

FEBRUAR 1985 ÖS 55,—/Sfr. 6,50 Lit 5.500/hfl 8,—/dkr 30,— DM 6,50

2|85

DAS MAGAZIN FÜR COMPUTER-FANS

Eine Markt & Technik Publikation

Ausführlicher Test
Was kann der Plus/4?

Vergleichstest
Assembler unter 100 Mark

Basic-Programme beschleunigen
Die besten Compiler im Vergleich

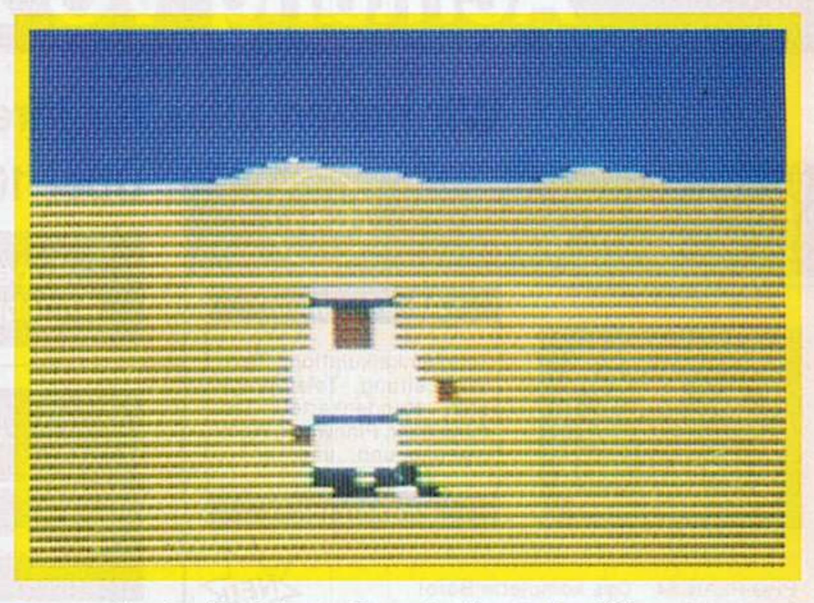
Abenteuerspiele
★ Spannend und in deutsch: Gordon Saga
★ Lösung von Hobbit

Listing des Monats
Das Grab des Pharao

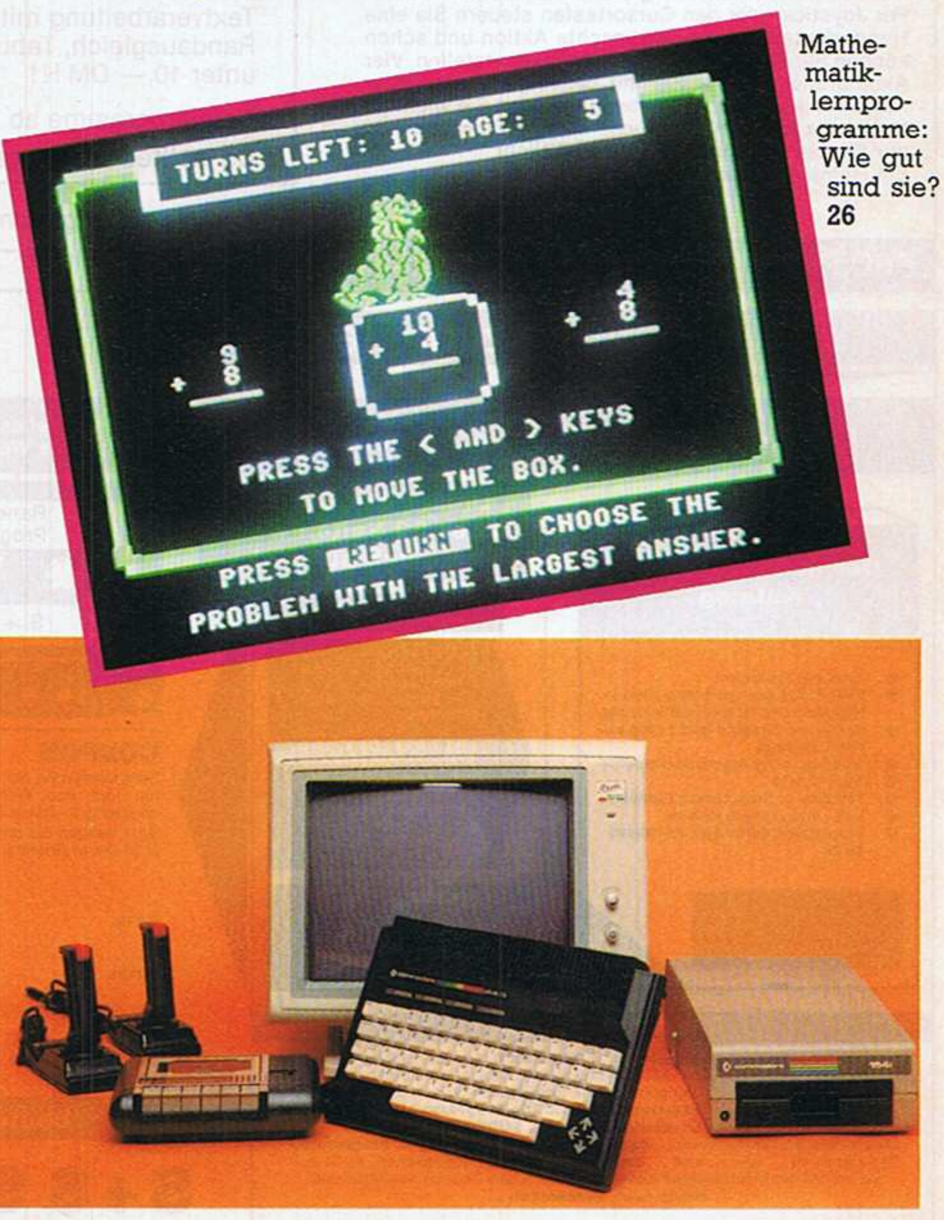
Verbindungskabel im Selbstbau
Fernseher wird zum Monitor

Alle Listings mit Prüfsummen
NEU
Nie wieder Fehler beim Abtippen

INHALT



Fragen Sie den Beduinen: Er weist Ihnen den Weg zum Listing des Monats **51**



Mathematiklernprogramme: Wie gut sind sie? **26**

Der neue Computer von Commodore: Was kann der Plus/4? **12**

Die eigene Stimme aus dem C 64 hören Voice Master macht's möglich! 19

Gordon Saga ist ein deutsches Abenteuerspiel, das ganze Sätze versteht 48



Notlandung als »lustiges Programm« 156

Listings zum Abtippen

Kurse

Rubriken



Gut versorgt

Deutschen Rückstand machte die Münchener Unternehmensberatung Dr. Höfner aus: Pro Kopf der Bevölkerung seien 1983 in den USA zehnmal soviel Mikrocomputer für gewerbliche Zwecke verkauft worden wie in der Bundesrepublik.

Bei Heimcomputern stehen wir wesentlich besser da: Bis Ende 1984 hatte Commodore nach eigenen Angaben in der Bundesrepublik rund 700 000 Heimcomputer verkauft — 100 000 VC 20 und 600 000 64er. In den USA wurde etwa das Dreieinhalbfache davon abgesetzt, rund 2 Millionen 64er und 1,1 Millionen VC 20. Geht man davon aus, daß die Vereinigten Staaten etwa die vierfache Bevölkerungszahl haben, und daß der Commodore-Marktanteil in beiden Ländern ungefähr gleich groß ist, dann kann man sagen: Die Deutschen sind mit Heimcomputern etwa gleich gut versorgt (oder gleich stark daran interessiert) wie die Amerikaner. Diese Schlußfolgerung läßt sich auch dann noch aufrechterhalten, wenn man berücksichtigt, daß in den USA häufig Systeme wie Apple II oder IBM-PC als »Homecomputer« verwendet werden.

Hier zeigt sich eine Tücke der Statistiken: Wir werden uns an neue Bedeutungen bekannter Begriffe gewöhnen müssen. Der Heimcomputer wird daheim, der Bürocomputer im Büro benutzt — über die Art und Leistungsfähigkeit des Grundgerätes sagen solche Bezeichnungen in Zukunft noch weniger aus als heute. Und da die Leistung der kleinen billigen Computer zunimmt (auch von Commodore soll es ja 1985 leistungsfähigere Modelle geben), wird künftig für viele berufliche Anwendungen ein »Heimcomputer« ausreichen. Damit werden sich die privat gewonnenen Erfahrungen noch sehr viel direkter als bisher auf den Computer im Betrieb anwenden lassen.

Michael Pauly, Chefredakteur

MCI Mail: die schnelle Post

In den USA gibt es private Postbeförderungsunternehmen mit einem erstaunlichen Angebot. Wir wollen am Beispiel der MCI Mail zeigen, was diese Firmen zu leisten imstande sind. Vor allem soll aber der C 64 über das Datex-P-Netz an diesem Service teilnehmen.

Briefe, genau genommen deren Laufzeit zum Empfänger, sind nicht nur hier in der Budesrepublik ein ständiges Ärgernis und ein Dauerthema in den Tageszeitungen. In den USA sind die (Post-) Wege naturgemäß noch länger und die staatliche US-Mail muß sich häufig vorwerfen lassen, eine »Snail-Mail«, eine Schnekkenpost zu sein. Und wenn es bei uns in Deutschland darum geht, ob ein 80-Pfennig-Brief nun wirklich am nächsten Morgen vom Briefträger ausgetragen wird oder nicht, dann geht es in Amerika darum, ob der Brief mit »Federal Express Overnight Delivery« für 14,00 Dollar (43 Mark) nun wirklich über Nacht von New York nach San Francisco gelangt.

Wen wundert es da, daß in den USA das Geschäft mit der privaten Postbeförderung blüht. In TV-Werbespots stechen sie sich gegeneinander aus, einer ist schneller und sicherer als der andere.

Den Vogel abgeschossen, und man könnte fast meinen, das Wappentier der USA sei damit gemeint, hat zweifellos ein Unternehmen, das schon seit Jahren mit ihren preiswerten Telefon-Fernleitungen der großen »Ma Bell« beziehungsweise der AT & T Konkurrenz macht: MCI. Diese Gesellschaft ist inzwischen so mächtig, daß sie eine große und bekannte Kommuniktions-Gesellschaft, die Western Union (WUI) mit ihren Telex-Netzen geschluckt hat. MCI hat also eine Tochtergesellschaft gegründet: die MCI Mail — The nation's new postal system.

Seit rund einem Jahr gibt es MCI Mail, und die Anzahl der Benutzer vermehrt sich geradezu nach dem Schneeball-System. Der Grund: die Postbeförderung von MCI braucht nur Sekunden! Das Geheimnis: elektronische Postfächer, also »Mailboxen« mit den Zusatzdienstleistungen eines riesigen Kommunikations-Apparates, seinen Datenleitungen, Druckerstationen, Telegrammboten und Telexnetzen. Und das Ganze zu erschwinglichen Preisen.

> In Ausgabe 10/84 von 64er hat Thomas Obermair über »Datex-P und ausländische Netzwerke« berichtet. Auf der Kenntnis dieses Artikels baut der folgende Bericht auf und setzt insofern Wissen über das Funktionieren von Datex-P, dem PAD, einer NUI und so weiter voraus.

Die jährliche Grundgebühr, die Annual Mailbox Fee, beträgt 18 Mark, auf den Monat umgerechnet also 1,50 Dollar oder rund 4,50 Mark.

Einen »Kurzbrief« von bis zu 500 Zeichen abzuschicken kostet 0,45 Dollar oder 1,35 Mark, ein Brief von bis zu 7500 Zeichen (etwa 4 Druckseiten) je 1 Dollar, also 3 Mark. Eine Speicherbenutzungsgebühr gibt es nicht. Es ist also völlig egal, wie lange der Brief im Speicher »lagert«. Das »Abholen« der Briefe aus dem Speicher ist kostenlos. In den meisten Städten der USA gibt es Telefonnummern, unter denen MCI Mail zum Ortstarif erreichbar ist. Wohnt man in ländlichen Gebieten, benutzt man eine sogenannte WATS-line, eine Telefonnummer mit der Vorwahl 800, bei der der Angerufene, also MCI Mail, die Gebühren übernimmt. Dieser Service ist dann allerdings mit 15 Cents pro Minute in der nächsten Monatsrechnung wiederzufinden.

Mit diesen Gebühren ist damit innerhalb der USA der ganz Spaß bezahlt. MCI Mail wäre aber kein typisch amerikanisches System, wenn es nicht zusätzlich eine Vielzahl von erstaunlichen Möglichkeiten böte, die der Bequemlichkeit ihrer Kunden entgegenkäme. Und zusätzlicher Service kostet natürlich Geld. Und das ist es, was man auch dort verdienen möchte, denn kein Postsystem der Welt, ob nun staatlich oder privat, arbeitet nur für den »Spaß an der Freud«.

Es fängt an mit dem Service »Mail Alert«. Für einen zusätzlichen Dollar ruft MCI Mail den Briefempfänger an, um ihn auf den Posteingang aufmerksam zu machen.

Und was ist, wenn der Briefempfänger nun gar kein »Postfach« bei MCI Mail hat? Kein Problem: Für einen Dollar mehr wird der Brief in der nächsten Großstadt ausgedruckt und mit der staatlichen Post als »First Class Mail« zugestellt. Das ist in der Regel nach 24 Stunden der Fall. Für 8 Dollar kann man sogar die Zustellung mittels Kurier am nächsten Tag verlangen, vorausgesetzt, der elektronische Brief wurde bis 23.00 Uhr eingespeichert. Und für ganz eilige Sachen gibt es noch den 4-HOUR-Service. Für sage und schreibe 30 Dollar, also 90 Mark wird die Zustellung in den ganzen USA per MCI-Mail-Boten in spätestens 4 Stunden garantiert.

Wie sehen diese gedruckten Briefe aus? Nun, im Normalfall, zum oben genannten Preis, handelt es sich um Briefpapier mit MCI-Briefkopf. Dazu wird Name und Adresse des Absenders gedruckt, dann Name und Anschrift des Empfängers, anschließend der Text und die Unterschrift in Druckschrift. Aber: Für 20 Dollar pro Jahr kann man auch seinen eigenen Briefkopf einspeichern lassen, der sogar grafische Darstellungen enthalten

kann. Und für weitere 20 Dollar wird sogar unter jeden in der Ferne gedruckten Brief Ihre Original-Unterschrift gesetzt!

Aber das ist noch nicht alles. Wenn der Brief weder in der Mailbox bleiben, noch ausgedruckt werden soll, kann man auch jede beliebige Telexnummer auf der Welt als Anschrift eingeben. Vollautomatisch wird der Brief dann als Telex abgeschickt und man erhält in der eigenen Box eine Nachricht, daß alles auch angekommen ist (mit der Kennung des erreichten Telex-Anschlusses als Beweis). Dafür muß man dann natürlich die Telexgebühren bezahlen, aber die sind verhältnismäßig niedrig. Je 400 Zeichen von den USA nach der UdSSR, kosten zum Beispiel 1,82 Dollar. Und unter jedes Telex schreibt das System die »eigene« Telex-Nummer, denn von jedem Telexanschluß in der Welt kann man Briefe an MCI-Mail schicken, die einfach in der jeweiligen Mailbox des Empfängers abgelegt wird und sofort abrufbereit ist. Als Telexnummer dienen dazu die Ziffern 650 und die siebenstellige Kennziffer (ID) des MCI-Kunden. Von Deutschland aus wählt man also inklusive der Vorwahl für die USA zum Beispiel: 023-650-2412526.

Soviel generell über das Funktionieren dieses Privatpost-Unternehmens MCI Mail in den USA. Meine Freunde in Amerika, Privatpersonen und Geschäftsleute, benutzen es täglich. Und es kam wie es kommen mußte: Man fragte mich, ob ich nicht auch zwecks schnellerer Kommunikation an MCI Mail teilnehmen könnte. Keine Frage: Ich wollte schon — aber wie? Ich schrieb also an MCI Mail, Box 1001, 1900 M Street, NW Washington, DC 20036, Tel. (00 12 02) 833 84 84, und erhielt auf meine Anfrage, ob ich von Deutschland aus an MCI teilnehmen könne, und wie das zu machen sei, die freundliche Antwort von einer Dame namens Alice J. Campbell, daß man eine »exciting news«, eine aufregenden Nachricht also, für mich hätte: Der Verkehr mit dem Ausland sei in einer Testphase. Neben der Mitgliedschaft bei MCI Mail müsse ich nur die örtliche PTT, also die Bundespost, um Zugang zu deren Datennetz bitten. Man wünschte noch viel Erfolg. Aus. Leider kein Hinweis, ob damit nun DATEX gemeint war und welche Nummern man wählen muß. Also weitergeforscht.

Inzwischen war mir zu Ohren gekommen, daß MCI und Western Union International irgendwie identisch seien. Und ich hatte mal gehört, daß WUI ein Büro in Frankfurt/M haben sollte. Die Telefonauskunft bestätigte dies und ich riskierte ein paar Einheiten. Ein freundlicher Herr mit gleichem Namen wie ich, hörte sich meine Wünsche geduldig an und hatte auch den ehrlichen Willen, mir zu helfen. Dennoch, über MCI Mail war nichts bekannt.

Keine Informationen bei der Post

Nun war das Leitungsmonopol der Bundesrepublik an der Reihe. Mein Brief an die Kundenberatung für Dateldienste beim Fernmeldetechnischen Zentralamt (FTZ) in Darmstadt wurde mit interessanten Drucksachen über DATEX und mit einem langen Telefonat beantwortet. Ein Kontakt, der übrigens noch heute besteht und dem sich manche interessante Neuigkeit entlocken läßt. Ein konkretes Ergebnis, etwa in der Form, welches Netz von Datex-P aus, also zum Beispiel WUI, RCA, ITT, TYMNET, TELENET und so weiter angewählt werden müsse, und wie dann die Teilnehmer-Nummer von MCI Mail angefügt werden könnte, ergab das Gespräch zu meinem Kummer nicht. Aber nur ein paar Tage später rief das FTZ nochmals zurück, mit einem interessanten Hinweis: Ich möge mich doch mal an die große Kommunikationsgesellschaft TYMNET wenden. Die hätten ein Europa-Büro in Paris! Gesagt, getan, ich wollte schließlich keine Chance auslassen. Gerechterweise muß ich hier einfügen, daß dieser Tip Gold wert war, und schließlich auf die richtige Spur führte. Die TYMNET-Mitarbeiter Dominique Marchad und Jean Francois Morfin kümmerten sich mit einer überraschenden Energie und einer Fülle von Nachrichten um mein Anliegen. Nicht von ungefähr, wie ich erfuhr: TYMNET ist verantwortlich für die internationale Aufschaltung von MCI Mail, und deren »host« 004759.

Also schnell nachgeschaut in der Datex-P Bedienungsanleitung der Bundespost: TYMNET hat die Vorwahl 03106. Dazu 004759, das müßte es also sein. Vorsorglich hatte ich mir bei der Post schon eine NUI besorgt, und nun konnte der Versuch mit meinem C 64, der 1541, dem Drucker Seikosha GB100VC und dem Tandy-Akustikkoppler AC-3 starten.

Die Hamburger Zugangsnummer für Datex-P war wie die anderer Großstädte der Datex-Bedienungsleitung zu entnehmen, also 441231. Sogleich ertönt der Carrier-Ton und ich gab vorschriftsmässig .(CR) ein. Nun wurde ich nach meiner NUI und dem von der Post zugeteilten Passwort gefragt. Das System antwortete, daß meine NUI nun »aktiv« sei — was natürlich vor allem bedeutet, daß man nun die anfallenden Gebühren berechnen würde. Ich tippte auf der Tastatur 03106004759. Nach zirka 5 Sekunden kam die Antwort: »Port 48 please log in«. Hier wurde nun der Name des MCI-Mitgliedes erwartet, und zwar in der Form, daß ein Herbert Schmidt eingeben müßte: HSchmidt/half. Das »half« ist notwendig, um den MCI-Computer auf Halbduplex zu stellen. Sonst gibt es Echo-Probleme. Danach wurde das von MCI Mail zugeteilte Passwort erfragt.

Früher, so wurde mir gesagt, konnte man als Neuling als Name und als Passwort jeweils einfach das Wort REGISTER eingeben und sich dann als neues Mitglied eintragen. Wenn man das heute versucht, antwortet einem der MCI-Computer höflich, daß die Registrierung »online« nicht mehr möglich sei. Man möge bitte anrufen. Das gilt natürlich für Kunden in den USA. Wenn Sie aber so viel Geld haben, können Sie gern die oben genannte Telefonnummer anrufen. Das kostet pro Minute 6,67 Mark. In Washington ist es übrigens 6 Stunden früher als hier.

In meinem Fall hat übrigens der freundliche Mr. Morfin von TYMNET, Paris, bei einem Besuch in Washington die Registrierung vorgenommen. Eine Woche später hatte ich einen der auffälligen, orangefarbenen MCI Mail-Umschläge in der Hand, genannt »starter kit«, mit genauen Eräuterungen und meinem Passwort. Das war's. Und sie waren zufrieden und gücklich bis ... bis zur ersten Rechnung?

Nun, halb so schlimm. Die 18 Dollar bei der ersten Rechnung tun vielleicht ein wenig weh. Aber das kommt ja nur einmal im Jahr. Und zwei Briefe pro Woche abgeschickt, sind 8 Dollar im Monat, also 24 Mark. Erwartet wird aber in jedem Fall Bezahlung per Scheck in US-Dollar und nicht per Euroscheck. Auf Antrag ist auch Abrechnung über eine Kreditkarte wie VISA oder MASTERCARD (hier EUROCARD) möglich. Dazu ein Tip: bei Banken und sogar bei der Post gibt es American Express Traveller Cheques zum Dollar-Tageskurs plus 1 Prozent Versicherung in Stückelungen bis herunter zu 10 $.

In den USA zahlt man Rechnungen grundsätzlich mit Scheck per Post. An jeder Rechnung aus Amerika hängt daher ein Abschnitt, den man mit seinem Scheck zurückschickt. Und da nimmt man von Deutschland aus am besten einen Traveller Scheck, mit dem nächsthöheren Wert, schickt das Ganze per Luftpost nach Washington und bei eventueller Überzahlung wird der Restbetrag bis zur nächsten Rechnung gutgeschrieben.

Und wie sieht es nun mit den Kosten für Datex-P aus? Nun, in der Kürze liegt die Würze. Wer seine Briefe erst online »komponiert« und die

Verbindung so lange stehen läßt, bis alles geschrieben ist und hübsch sauber aussieht, kann schon mal anfangen Geld zurückzulegen, damit die Rechnung der Post keine zu große Überraschung bringt. Wenn man aber ein gutes Terminalprogramm besitzt, wie zum Beispiel das TERM 64 von Higginbottom für den C 64, dann kann man Briefe als Files vorschreiben, auf Diskette speichern und dann erst die Verbindung zu MCI Mail herstellen. Die Übertragung geht dann mit Höchstgeschwindigkeit, also 300 Baud vor sich. Umgekehrt gibt es zum Auslesen der in der Mailbox etwa vorhanden Briefe einen Befehl »PRINT INBOX«. Dann kommen alle Briefe in Höchstgeschwindigkeit und ohne Unterbrechung hier an, gehen zunächst in den Terminal-Puffer, und können dann in Ruhe nach Auftrennen der Verbindung auf Floppy geladen, auf dem Bildschirm gelesen und/oder ausgedruckt werden.

5 Minuten Verbindung mit, sagen wir 4000 Zeichen, kosten bei Datex-P neben der monatlichen NUI-Gebühr von 15 Mark und der Telefon-Ortsgesprächsgebühr von 23 Pfenningen nach den USA:

1. Zugangsgebühr	DM 0,20
2. Zuschlag je Verbindung	DM 0,05
3. Anpassungsgebühr je PAD-Benutzung	DM 0,30
4. Zeitgebühr USA	DM 1,00
5. Volumengebühr USA	DM 1,00
Zusammen also	DM 2,55

Datex-P-Gebühr von 2,55 Mark und MCI-Mail-Gebühr von 3 Mark ergeben also 5,55 Mark für den Brief von 4000 Zeichen... und das ist eine ganze Menge Geschriebenes. Und mit einem kleinen Trick kann man seinen Partner in den USA schon wenige Sekunden danach veranlassen, seine Mailbox zu lesen: Man ruft ihn an, für zwei Einheiten, also 46 Pfennige, kann man schnell die Worte »MCI Mail« sagen und wieder auflegen. Das wirkt garantiert. Der Brief ist somit wenige Minuten nach dem Absenden gelesen... für runde 6 Mark. Ein Luftpostbrief, per Eilboten, 2 DIN-A-4-Seiten lang, dauert mit Sicherheit 5 Tage... mindestens... und kostet (18 Gramm) 5,50 Mark. Was sagen Sie nun?

Und kürzlich gab MCI Mail noch eine Neuerung für 1985 bekannt: Es wird der Postverkehr mit 40 Ländern aufgenommen! Und das bedeutet nicht nur elektronischen Zugang wie zur Zeit aus Deutschland, sondern auch Ausdruck und Zustellung der Post in den entsprechenden Ländern. Vielleicht kann man Briefe also ebenso schnell und preiswert nach Australien oder Südafrika schicken. Wenn das nichts ist? Aber das Schönste dabei ist doch: Es funktioniert alles ohne Probleme mit einem Commodore 64.

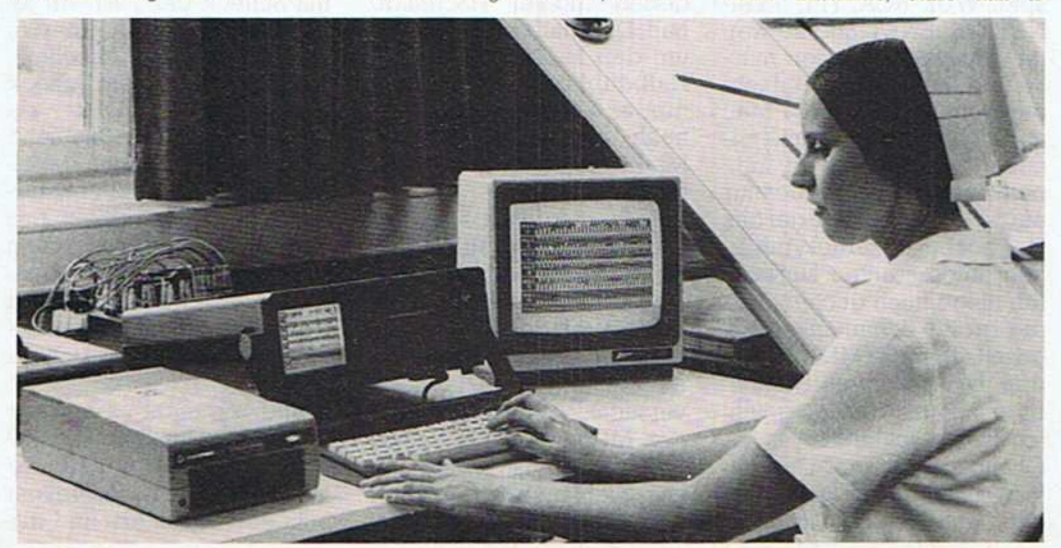
(Wolfgang R. Schulz/aa)

SX 64 spart dem Krankenhaus viel Geld

In einem großen Krankenhaus mit einer Vielzahl weiterer angegliedeter Pflege-, Schulungs- und Wohnungseinrichtungen fallen hohe Investitions- und Unterhaltungskosten an. Rationelle Betriebsführung anhand einer zuverlässigen Betriebsdatenerfassung, führt zur erheblichen Kosteneinsparung. Eine Sache für die Groß-EDV, die selbst Millionen kostet? Keineswegs: Im Diakonie-Mutterhaus Rotenburg an der Wümme, schafft ein tragbarer Commodore SX 64 die Voraussetzung dafür. Allein die Ersparnis an Wasser beträgt im ersten Jahr schon 50000 Mark.

Dabei hatte Betriebsingenieur Ulrich Hartnick hauptsächlich die Sammlung von Betriebsdaten im Sinne, um für Neuanschaffungen im Energie- und Versorgungsbereich vernünftige Planungsdaten zu gewinnen. So soll zum Beispiel die in der Vielzahl der Kühlanlagen abgezogene Wärme zur Vorwärmung des Brauchwarmwassers verwendet werden. Dazu muß die abgezogene Wärmemenge und die Menge des verbrauchten Wassers bekannt sein. Auf dieselbe Weise müssen für jede geplante neue Einrichtung erst die genauen Bedarfsdaten ermittelt werden. Das Erfassen und Verarbeiten solcher Daten in Zusammenarbeit mit den entsprechenden Meßgeräten ist die Aufgabe des SX 64.

An sechs verschiedenen Stellen fragt der SX 64 die durchgelaufenen Wärmemengen ab. Viertelstündlich werden die Wassermengen notiert, die Daten auf Diskette abgelegt und auf Wunsch auf dem Bildschirm oder dem Drucker ausgegeben. Über eine programmierte Kalenderfunktion hält der SX 64 Monat, Tag und Uhrzeit zu jeder Messung fest. Die Meßwerterfassung erfolgt über eine einfache Ein-Aus-Abfrage an Magnetschaltern und über Haustelefon. Das verwendete Interface stammt von GSSE in Braunschweig. Das Auswertungsprogramm wurde von Ulrich Hartnick selbst erstellt. Eine Analyse des Wasserverbrauchs führte zur Identifikation ständig laufender Wasserverbraucher, die aber prinzipiell nur kurzzeitig benötigt werden. Das Abstellen dieses »Dauerlaufs« senkte den Wasserverbrauch um etwa ein Viertel — so kommt die jährliche Einsparung von etwa 50000 Mark zustande, ohne daß irgendeine Funktion des Krankenhauses im geringsten beeinträchtigt wird.



Der Hauptvorteil liegt jedoch darin, daß endlich der Bedarf an Energie und Versorgungsmitteln zu jeder Tages- und Nachtzeit genau ermittelt werden kann. Der SX 64 wertet die Ergebnisse aus, ein weiterer Computer ist nicht erforderlich.

Insgesamt zeigt der Einsatz des SX 64 am Diakonie-Krankenhaus in Rotenburg neue Chancen für eine umfassende Betriebsdatenerfassung auf. Überall existieren schon große und leistungsfähige Computer für kaufmännisches Rechnungswesen, Personalkosten und so weiter. Aber im täglichen Betrieb nutzen sie, und seien sie noch so teuer, nur selten etwas: Ihnen fehlen die entsprechenden Daten. Erst die Datenerfassung an der Basis macht den Betriebsablauf transparent, hilft Leerläufe und Verschwendungen erkennen und schafft verständliche Unterlagen für die Planung und Entscheidung. Eine Integration der am Commodore SX 64 oder auch am C 64 gewonnenen Daten in die Groß-EDV, ist sicherlich möglich und bietet auf längere Sicht die Fähigkeit, ein beliebiges Unternehmen ohne wesentlichen Leerlauf, Verschwendung und unter Verwendung der rationellen Arbeitsmittel zu führen. Der Weg ist vorgegeben, weg von der zentralen Lösung, hin zur dezentralen Erfassung von Daten mittels preisgünstiger »Heimcomputer«. (aa)

Info: Commodore, Lyoner Str. 38, 6000 Frankfurt/M. 71, Tel. (069) 6638-0

Leser fragen — Willi Brechtl antwortet

Mein Name ist Willi Brechtl. Nachdem mein Cousin, das Fehlerteufelchen, hier soviel Unheil angerichtet hat, habe ich mir gedacht, daß ich etwas Gutes tun kann — sozusagen als Ausgleich.

Ich werde mich daher um Leserbriefe kümmern, die nicht in das sachliche Einerlei des Leserforums passen. Zum Beispiel Fragen, die sich aus dem einen oder anderen Grund nur ganz subjektiv beantworten lassen. Oft genug tauchen auch Probleme auf, die sich nicht mit einem kurzen Antwortsatz abhandeln lassen. Und wenn selbst eine längere Antwort im Rahmen des Leserforums nicht mehr ausreichen würde, dann ist das ganz klar ein Fall für Willi Brechtl.

Natürlich habe ich mir zur Beantwortung das erste Mal auch gleich entsprechende Fragen herausgesucht. Möglicherweise sind einige Fragen darunter, über die viele Profis schmunzeln können, aber meine Beantwortung ist — so hoffe ich — immer fair genug.

Kleine Bösartigkeiten mögen mir dabei verziehen werden (ich kann meine Verwandtschaft zum Fehlerteufelchen eben doch nicht leugnen).

Also: Wenn Sie als Anfänger Probleme mit Computer, Software oder Handbuch haben, dann wenden Sie sich in Zukunft doch vertrauensvoll an mich.

Maschinensprache sprechen?

Ich will mit meinem C 64 RICHTIGE Maschinensprache sprechen und finde kein Trainingsbuch dazu. Ich will keinen Assembler! Wer kann mir da helfen?

Meine definitive Antwort lautet: Keiner. Assembler ist nun mal Maschinensprache und Maschinensprache ist Assembler.

Das will ich Ihnen so erklären: Unter Assembler verstehen Sie wahrscheinlich Befehle wie LDA (Lade Akku) oder CMP (Vergleiche). Maschinensprache ist dann die Form, in der das Programm im Speicher liegt, beispielsweise als $CD für Vergleich oder $A9 für Akku laden. Aber auch das ist noch nicht die eigentliche Maschinensprache, denn die besteht aus Folgen von Nullen und Einsen.

Um aber diese Maschinensprache zu sprechen und in hex oder sogar binär zu programmieren, kommen Sie nicht umhin, die Assemblerbefehle zu lernen — oder wollen Sie wirklich auswendig wissen, daß »11001101« ein Vergleichsbefehl ist? Na sehen Sie. Und um Assembler (Maschinensprache) zu lernen, dafür gibt es genügend Trainingsbücher (und auch unseren 64'er-Assemblerkurs).

Wieviel Strom braucht der C 64?

Welche Stromkosten verursacht ein einen Monat lang ununterbrochen laufender C 64? Bitte in DM und als Vergleich (zum Beispiel »soviel wie eine 100-Watt-Glühbirne«).

Gegenfrage: Haben Sie Ihren C 64 etwa einen Monat lang ununterbrochen laufen lassen? Und wenn ja, haben Sie etwa nebendran zum Vergleich eine 100-Watt-Glühbirne brennen lassen?

Also: Der C 64 hat genau 15 Watt Leistungsaufnahme (steht jedenfalls auf dem Computer). Die elektrische Arbeit in kWh Kilowattstunden bemißt sich aus Leistung mal Zeit. Wenn Sie das Floppy-Laufwerk auch noch mit einbeziehen (50 W), haben Sie einen Verbrauch von 46,8 kWh im Monat. Das einzige, was Sie noch tun müssen, ist, Ihr Elektrizitätswerk nach dem Preis einer kWh zu fragen. Rechnet man grob etwa 20 Pfennige pro kWh, dann kostet der Dauerbetrieb von Computer und Floppy noch keine 10 Mark im Monat. Allerdings ist darin noch nicht der Stromverbrauch des ja wohl in den meisten Fällen ebenfalls eingeschalteten Monitors oder Fernsehgerätes berücksichtigt.

Ungeöffnete Disketten anwenden?

Die Disketten, die ich besitze, kann ich nicht anwenden, weil sie nicht geOPENt sind, deshalb bitte ich um Vorschläge, wie man auf Disketten eine sinnvolle Datei eröffnet.

Ohne Ihnen zu nahe treten zu wollen schlage ich vor, daß Sie sich das Floppy-Handbuch mal durchlesen (so schlecht ist es ja nun auch wieder nicht!). Dort finden Sie den deutlichen Hinweis, daß man die Diskette vorher FORMATIEREN muß, um sie benutzen zu können. Und ob die Datei sinnvoll ist, das hängt von Ihnen und nicht vom Datenträger ab.

Lötkolben als Interface?

Wie kann ich mittels Lötkolben eine elektrische Schreibmaschine am C 64 betreiben? Könnte man die Treibersoftware dazu auf ein EPROM brennen?

Natürlich kann man jede Software auch auf Eprom brennen (wer sollte einen daran hindern?). Allerdings müssen Sie sich dann selbst eine entsprechende Platine zum Einstecken in den Steckmodul-Port basteln.

Das Betreiben einer elektrischen Schreibmaschine wird aber mit einem Lötkolben alleine nicht funktionieren. Vielmehr steuert man solche Dinge mit einem INTERFACE an.

Wenn Sie sich allerdings selbst ein Interface bauen wollen, kann ich Ihnen leider auch nicht weiterhelfen. Denn dabei kommt es ganz auf Ihre Schreibmaschine an, beziehungsweise auf deren internen Aufbau.

Spektakuläre Verbindung

Wie kann ich einen ZX 81 und einen C 64 miteinander verbinden, und welchen Vorteil habe ich davon?

Nun, das mit dem Vorteil kommt ganz darauf an, aus welcher Sicht Sie das Problem sehen. Versetzen Sie sich einmal in die Lage des ZX 81 . Dann haben Sie natürlich gewaltige Vorteile davon, weil Sie durch den Anschluß an den C 64 zum Supermann geworden sind. Aber denken Sie doch mal an den C 64! Was soll der mit einer als Computer getarnten Plastikkarte schon anfangen?

Nun aber mal ernst: Verbinden kann man beide Geräte durchaus. Aber Vorteile fallen mir dazu keine ein. Die Hauptspeicherkapazität des ZX 81 beträgt exakt 1024 Byte, wovon noch der Bildschirmspeicher abgeht. Bevor ich also Daten vom ZX 81 an den C 64 sende, tippe ich sie lieber gleich am C 64 ein, das geht schneller.

Stiftung Warentest zum Thema Heimcomputer

Antwort eines Lesers auf den Test der Heimcomputer in der Oktoberausgabe von Stiftung Warentest.

Bemerkenswert schlecht kamen/kommen die Heimcomputer im Bericht und Test, Ausgabe 10/84 weg.

Aber die Heimcomputer haben es nicht verdient, denn die Redaktion von »Test« ging mit der falschen Fragestellung an diese Geräte heran. Es handelt sich nicht um Waschmaschinen oder Toaströster und auch nicht — trotz magnetischer Aufzeichnung und digitaler Technik — um Videorecorder oder CD-Plattenspieler. Beim Angeln oder Drachenfliegen fragt man so wenig nach der Nutzanwendung wie beim Kakteenzüchten oder Briefmarkensammeln.

Die schlechte Meinung über praktisch alle Geräte kann man gleichwohl weitgehend teilen — bis auf den Commodore 64.

Es wurde nach konkreter Nutzanwendung gefragt? Bitte: Beispiel 1: Ich nutze meinen Commodore 64 unter anderem als recht komfortables Textver- und bearbeitungssystem privat und semiprofessionell. So entstehen private Briefe ebenso, wie Artikel für Fachzeitschriften mit Bildschirm- und Computerhilfe. Die Investition (Computer, Floppy-Laufwerk, Farbmonitor, Textsystem, Schreibmaschine mit Interface) von etwa 3500 Mark hat sich so durch Honorare schon fast bezahlt gemacht.

Beispiel 2: Mein Sohn begann in Englisch bei den wöchentlichen Vokabelarbeiten abzurutschen: erst eine 3, dann 4, dann 5. Jetzt wurde der 64er mit einem Vokabel-Lern- und Übungsprogramm gefüttert und fleißig geübt. Das Ergebnis: nach der 5 kam eine 1! Gewiß, auch durch konventionelles Pauken hätte sich so was sicher erreichen lassen, aber mit weniger Spaß und mit sehr viel mehr Einsatz der beiden berufstätigen Eltern.

Und noch ein positiver Aspekt für einen Heimcomputer, wenn man Kinder hat: die wachsen mit einem Instrument wie selbstverständlich heran, das in vielfältigen Formen und Aufgabenstellung ihren künftigen beruflichen Alltag bestimmen wird. Wenn wir Erwachsenen an die zahllosen Probleme denken, denen wir im Beruf bei der Einführung der EDV an unseren Arbeitsplätzen ausgeliefert sind, so kann man die Jugend zumindest in diesem Punkt beneiden, weil sie ohne Hemmungen an diese neue Alltagstechnik herangeht.

Und ein letzter Punkt: Heimcomputerei ist etwas für kommu-

nikationsfreudige Menschen. Das scheint zunächst widersprüchlich zu sein, wenn man die emsig, aber einsam vor ihren Bildschirmen Tippenden sieht. Nach meinen Erfahrungen finden sich aber bei der Computer-Gemeinde auffällig viel Leute mit Btx-Vergangenheit oder mit CB-Funk-Erfahrung. CB-Funk ist fast tot, Btx auch, es hat nur noch keiner gemerkt — auch die Btx-eifrige Stiftung Warentest nicht. Es sollten im Herbst 150 000 Btx'er sein: tatsächlich sind es erst 14 000. Der größte Teil davon dürften Anbieter sein oder Ex-Feldversuchsteilnehmer, die mit dem Post-Lockangebot (1 000 Mark) umstiegen — so wie ich.

Aber es gibt Hunderttausende von Heimcomputern, die leicht miteinander kommunizieren könnten — und viele tun es auch schon. Da gibt es Mailboxen, schwarze Bretter und viele interessante Informationen. »Schau'n Sie doch mal rein«, aber nicht Btx ist gemeint.

Ich glaube, daß sich Btx zum größten Kommunikations-Flop entwickelt. Aber das macht nichts, das merkt ja keiner, weil man das aus den horrenden Gewinnen des größten deutschen Monopolisten finanziert. Wir alle zahlen die Rechnung. Und damit klebe ich eine halbe Mark auf den Umschlag dieses Briefes, der natürlich auch am Monitor-Bildschirm meines Commodore 64 entstand.

(Klaus-Dieter Wüstermann)

Eine Bitte an alle Mailbox-Benutzer

Nachdem wir nun TECOS fast 7 Monate betreiben, hat uns Ihr Vergleich in der Ausgabe 12/84 mit dem CB-Funk gefallen, es deckt sich mit unseren Erfahrungen! Allerdings vor allem im Negativen: Unter dem Mantel der Anonymität werden Obszönitäten, Beleidigungen und Schwachsinn eingegeben.

Aus diesen Gründen ist TECOS sehr restriktiv geworden. In der Blockzeit zwischen Programmstart (meist 20h) und Mitternacht kann TECOS nur von PTC-Mitgliedern und eingetragenen Benutzern benutzt werden, Gäste erhalten einen entsprechenden Hinweis. Die Benutzerzeit für Gäste ist sehr gering (10 Minuten) und die Auswahl ebenfalls (nur 4 von 8 Punkten im Hauptmenü). Aber mit all diesen Sachen kann man als Box leben, wenn das Programm entsprechend reagiert und die Eingaben erst einmal »zensiert« (besser ausgedrückt: »gesichtet«) werden.

Aber ein viel größeres Problem sind nach unserer Ansicht die vielen »Hacker-Lehrlinge«, die sich einfach nicht an gewisse Betriebszeiten halten können und manchmal eine elende Plage sind. Vor allem, wenn sich außerhalb der Betriebszeiten einmal der SysOp (oder sonst jemand) »via Voice«, also per Sprache meldet. Wir haben seit einiger Zeit einen kleinen Artikel darüber mit folgendem Wortlaut in unserer Box stehen:

Wie können Hacker Geld sparen?

Lassen Sie uns mit einem Tatsachenbericht beginnen:

Wir als PTC sind jeden Abend zwischen 17 und 20 Uhr via Voice für unsere Mitglieder erreichbar. In dieser Zeit erhalten wir zirka dreißig Anrufe, die wir in drei Kategorien unterteilen:

A) Der verschämte Aufleger

Nach unserer Begrüßung via Voice: »PTC — TECOS, guten Abend« knackt es in der Leitung — aufgelegt. Zirka 20 Sekunden später das gleiche Spiel.

Warum fragt der gute Mann nicht einfach, ob da eine Box dran ist? Entweder hat er eine falsche Nummer, dann kann er das Spielchen morgen, übermorgen und so weiter wiederholen. Oder es handelt sich um eine Box, die nur zeitweise sendet, dann kann er sich die Anrufe vor 20h sparen!

B) Der superschnelle Carrier-Sender

Nach dem Aufnehmen des Hörers knallt uns schon der Carrier entgegen und verstummt, meist automatisch, nach 20 bis 30 Sekunden. Alle Versuche, hier jemanden anzusprechen, sind witzlos. Deswegen legen wir bei Anrufen mit Carrier außerhalb der Sendezeit von TECOS sofort wieder auf. Zirka 20 Sekunden später erfolgt meist ein Anruf nach Typ A

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines guten Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere, bessere Antwort als die hier gelesene, dann schreiben Sie uns. Antworten publizieren wir in einer der nächsten Ausgaben. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.



C) der unheimliche Entschuldiger

Man nimmt den Hörer ab und bevor man noch ein Wort herausgebracht hat, hört man: »Entschuldigung, falsch verbunden«. Manchmal klappt es noch, dem Anrufer ein »Mailbox erst nach 20 Uhr« entgegenzuschleudern, falls er dann noch nicht aufgelegt hat, hört man ein »Ach so, wußte ich nicht« und dann erst »CRCK«.

Falls das mit dem Entgegenschleudern nicht klappt, meist nach 20 Sekunden ein Anruf, Typ A!

Nun, nicht nur, daß derartige Anrufe derart geballt den freundlichsten SysOp verärgern, Sie schmeißen doch auch dem GILB das Geld in den Rachen und sind — wenn Sie sich so blöd verhalten — mit Ihrer Mailboxliste nie UP TO DATE!

Denn wenn es heute nicht klappt, versuchen Sie es doch morgen oder übermorgen wieder und schenken dem GILB wieder ein, zwei Einheiten!

Ist denn das so schwer, erst einmal kurz in jede Verbindung reinzuhören und wenn sich jemand mit Stimme meldet, einmal nachzufragen? Oder sind die Leute (Typ A-C) schon zu blöde zum Sprechen? Könnt Ihr denn nur noch Tasten kloppen? Also, versucht's mal auf die andere Art! Zufriedene Mienen danken es Ihnen!

(Dietmar Severitt, SysOp TECOS, PTC)

Simons Basic und Turbotape

Kann man Turbotabe und Simons Basic zusammen benutzen?
Ausgabe: 8/84
Ausgabe: 12/84 *Rudolf Lehr*

Es gibt jetzt eine Version von Simons Basic, bei der Turbotape integriert ist. Außerdem sind in dieser Version ein Monitor und ein Disassembler in Simons Basic enthalten.

Mailbox mit C 64

Wer kann mir Telefonnummern nennen, unter denen ein C 64 eine Mailbox bedient?
Ausgabe 12/84

Stephan Prinz

Die FIB (069-726527) wendet den C 64 als Infobox an. Ab Januar 1985 ebenso der User-Club Asperg + Tamm + Rest der Welt. Um eine Mailbox mit dem C 64 zu betreiben, braucht man auf jeden Fall ein größeres Laufwerk, zum Beispiel eine 8250 von Commodore, da sonst mit der 1541 nur zirka 40 Mailboxbenutzer mit Passwort und Mails (senden/empfangen) verwaltet werden können.

Steffen Gebauer, Asperg

Wo gibt's Lightpens?

Können Sie mir die Adresse einer Firma mitteilen, wo ich ein Lightpen plus Software (anschlußfertig an den C 64) für zirka 50 Mark bekommen kann?

Markus Lucassen, Duisburg

List und Löschschutz

Ausgabe 12/84, Seite 85

Den im Artikel beschriebenen Listschutz kann man folgendermaßen leicht knacken:

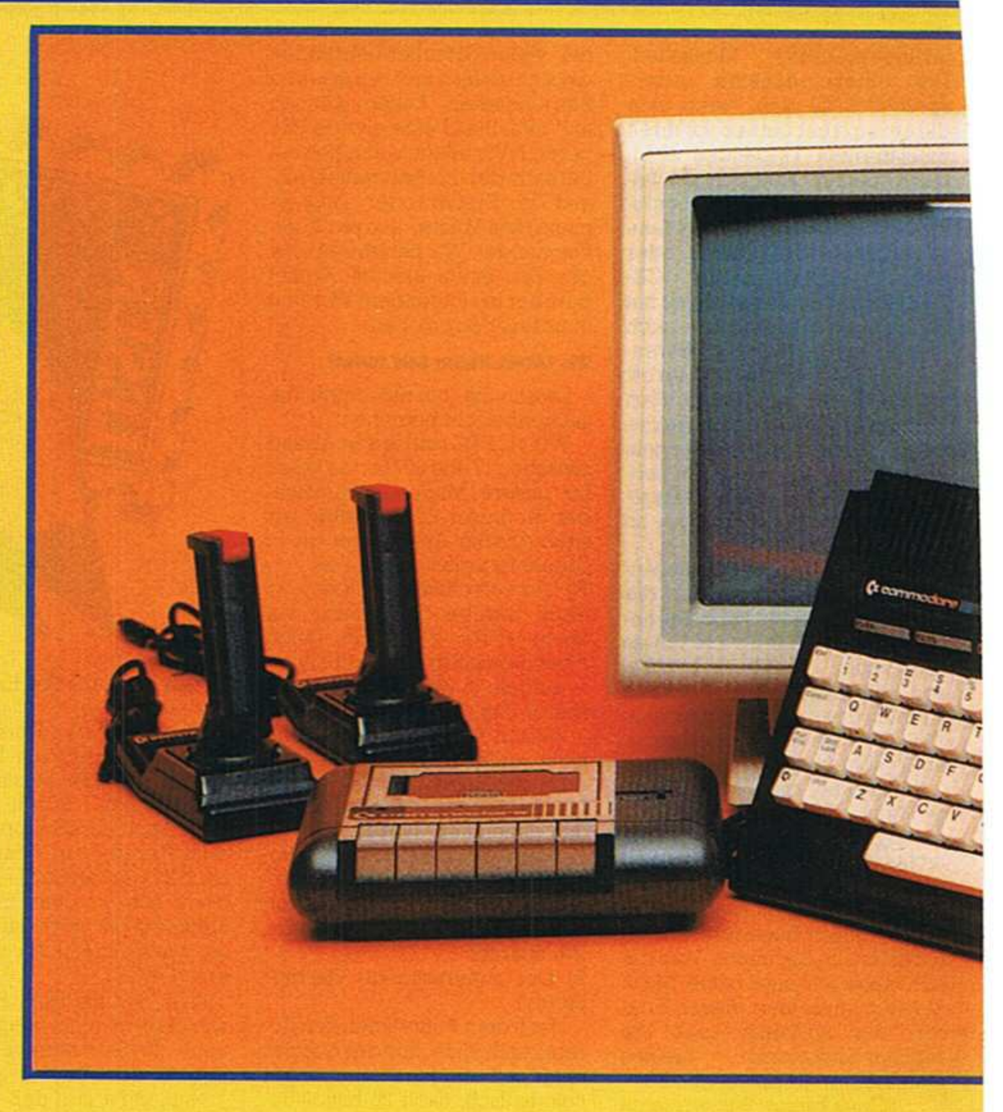
POKE2052,0:POKE2078,0

Jetzt kann man das Programm schon listen. Nun noch 0 (Return) und 1 (Return) eingeben, um die ersten beiden Zeilen zu löschen und das Programm ist wieder vollkommen ok.

Ulrich Förster, Bückeburg

Von zwei Seiten sollte man den neuen Plus/4 von Commodore betrachten. Auf der einen Seite steht die Hardware und das Basic, auf der anderen die eingebaute Software.

Ein großes Plus verdienen die Hardware und das Basic des Plus/4. Dem Benutzer steht, wie bei dem C 16, mit der Version 3.5 ein sehr gutes Basic zur Verfügung. Da in dem C 16-Testbericht (Ausgabe 1/85) schon ausführlich auf diese neue Basic-Version eingegangen wurde, soll hier nur noch einmal auf spezielle Eigenschaften hingewiesen werden.



PLUS UND MINU

Computer mit integrierter Software heißt das neue Konzep Computer mit diesem Konzept. Der Weg, den Commodor diesen Computer jedoch ein Preis von 1 298 Mark gerec

Basic 3.5 — Das beste Commodore-Basic

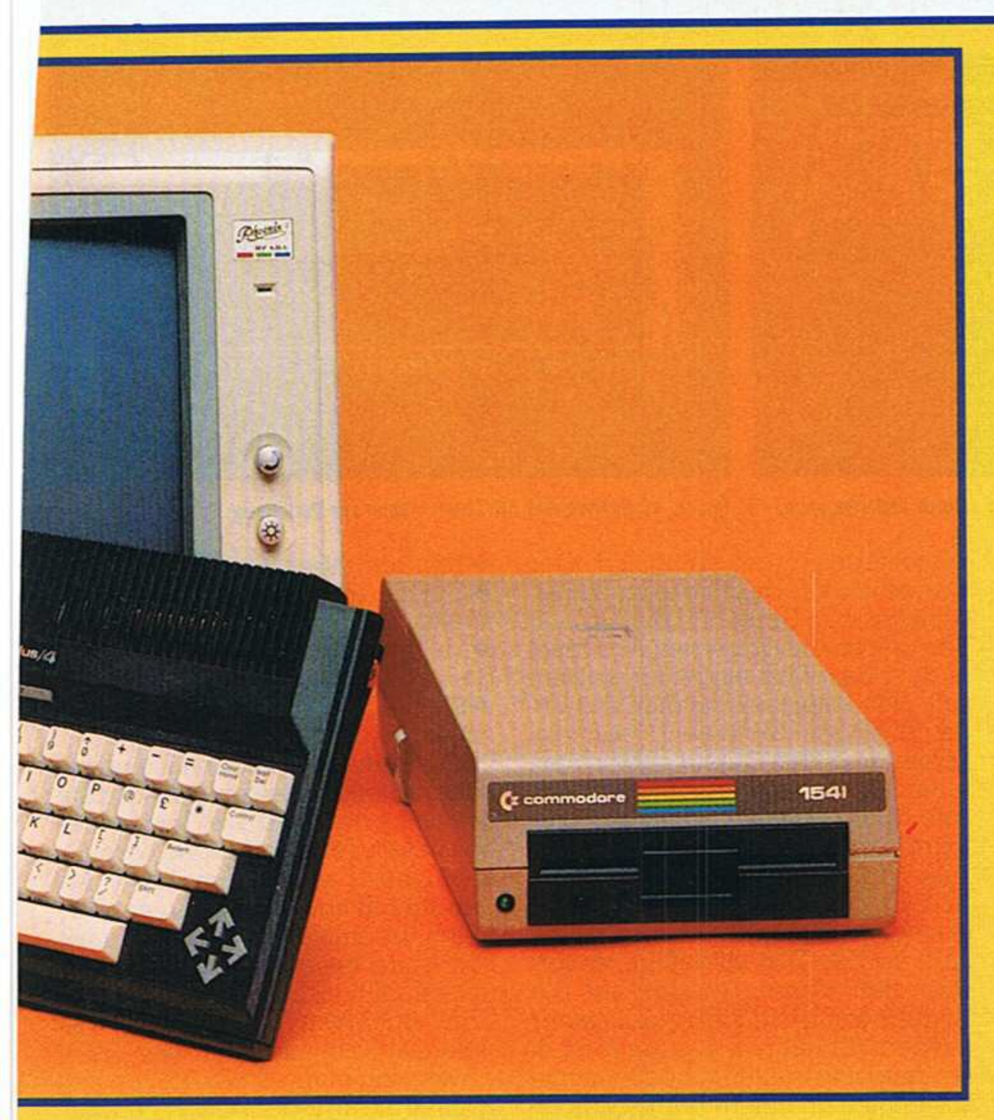
Mit der Basic Version 3.5 steht dem Programmierer ein gutes Werkzeug, und mit 60 KByte freiem RAM auch genügend »Spielraum« zur Verfügung. Befehle, mit denen man DO WHILE- und DO UNTIL-Schleifen aufbauen kann, unterstützen das strukturierte Programmieren. Das umständliche »herumPOKEn« für Grafik- und Tonerzeugung ist durch eine Reihe von leistungsfähigen Befehlen ersetzt worden. Einfache Grafiken können so mit wenigen Befehlen schon nach kurzer Einarbeitungszeit mit dem Plus/4 erstellt werden. Mit dem folgenden kleinen Programm kann man sehr einfach eine interessante Grafik erzeugen.

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC1,1
40 A = RND(1)*20+10
50 FOR L=0 TO 359 STEP A
60 CIRCLE 1,160,100,80,40,,,L
70 NEXT L
```

Sprites sind beim Plus/4 nicht zu finden. Dafür gibt es aber die »SHAPEs«. Mit den Befehlen SSHAPE und GSHAPE werden Ausschnitte aus dem Grafikbildschirm (Mehrfarben- oder Hires-Modus) als Basic-Stringvariable gespeichert beziehungsweise geladen.

Die Fehlersuche erleichtern zwei Funktionen. Zum einen ist die vierte Funktionstaste mit einem HELP-Statement belegt. Tritt im Programm ein Fehler auf, und drückt man diese Taste, erscheint die fehlerhafte Zeile auf dem Bildschirm und blinkt ab dem Befehl, bei dem der Fehler auftrat. Zum anderen stehen die Befehle TRON und TROFF zur Verfügung. Fügt man in ein Programm den Befehl TRON ein, so protokolliert der Plus/4 ab dieser Stelle den Ablauf eines Programmes. Jede Anweisung wird ausgeführt und die Zeilennummer, in der die Anweisung steht, auf dem Bildschirm ausgegeben. Der Befehl TROFF hebt diesen Status wieder auf.

Innerhalb eines Programmes läßt sich gezielt auf Fehlersituationen mit dem Befehl TRAP reagieren. Die Befehlssyntax lautet »TRAP (Zeilennummer)«. Tritt im Programmlauf ein Fehler auf, wird in die bei dem TRAP-Befehl angeführte Zeilennummer verzweigt. In der Variablen »EL« ist die Zeilennummer der Zeile abgelegt, in der der Fehler auftrat. In der Variablen »ER« ist die Fehlernummer gespeichert. Mit der String-Funktion »ERR$(ER)« läßt sich die Fehlermeldung auslesen und in der TRAP-Programmroutine verarbeiten. So kann man gezielt Fehler abfragen und auf sie reagieren, ohne einen Programmabsturz hervorzurufen. Ausgenommen sind hiervon nur der UNDEF'D STATEMENT ERROR und Fehler in der TRAP-Routine selbst. Mit der Anweisung



über einen Maschinensprache-Monitor mit dem Namen TEDMON. TEDMON ist allerdings nicht nur ein Monitor, sondern in ihm ist auch noch ein Disassembler und ein Assembler enthalten. Die Erstellung von Maschinenprogrammen ist mit dem TEDMON sehr komfortabel. Der Befehlssatz ist mit dem Monitor des C 16 identisch und wurde in der Ausgabe 1 bereits veröffentlicht.

Doch nun zu dem, was den Plus/4 am stärksten von anderen Computern abhebt.

Das neue Konzept

Der Plus/4 verfügt über eine eingebaute Textverarbeitung, eine Tabellenkalkulation, eine Dateiverwaltung und eine »grafische« Auswertung. Diese vier Programme haben dem neuen Computer seinen Namen gegeben. Faßt man das Plus im Namen als Additionszeichen auf, so kann man dem zustimmen. Alle vier Programme sind vorhanden. Als Wertung sollte man dieses Plus allerdings nicht sehen.

Alle vier Programme sind in einem 32 KByte-ROM untergebracht. Die Textverarbeitung kann mit den anderen drei Programmen Daten austauschen.

In die eingebaute Software gelangt man über die F1-Taste. Diese bringt einen SYS1525-Aufruf auf den Bildschirm, den man nur mit RETURN zu bestätigen braucht, um in die Textverarbeitung zu gelangen (Bild 1). Die anderen drei Programme kann man nur von hier anwählen. Will man das eingebaute Softwarepaket wieder verlassen, muß man den neben dem Netzschalter angebrachten Reset-Knopf betätigen. Ein Befehl zum Verlassen der Software ist nicht vorgesehen.

S BEIM PLUS/4

von Commodore. Der Plus/4 ist der erste Commodore- nit dem Plus/4 beschreitet, hat sicher Zukunft. Ob für rtigt ist, muß sich erst noch erweisen.

»RESUME« kann der Programmlauf wieder aufgenommen werden.

Das Bild im Bild

Mit dem Plus/4 ist es möglich, Teile des Gesamtbildschirms als »WINDOW« zu definieren. Ein »WINDOW« ist ein Ausschnitt des Bildschirmes, in dem von der Programmeingabe bis zur PRINT- oder Zeichenanweisung alles ablaufen kann, ohne den Bildschirminhalt außerhalb des »WINDOWS« zu beeinflussen.

Die Window-Technik des Plus/4 ist, verglichen mit der des Schneider CPC 64 etwas enttäuschend. So sind Windows nicht über direkte Anweisungen, sondern nur über die ESC-Funktionen zu erreichen. Weiterhin können nicht mehrere Windows gleichzeitig auf dem Bildschirm dargestellt werden.

Der Monitor

Für die Maschinensprachefreunde hat der Plus/4 einen besonderen Leckerbissen bereit. Er verfügt

Plus 1 — Die Textverarbeitung

Ein mit der eingebauten Textverarbeitung erstellter Text kann bis zu 99 Zeilen mit 77 Zeichen pro Zeile enthalten. Auf dem Bildschirm werden allerdings nur jeweils 22 Zeilen und 37 Spalten angezeigt. Schreibt man über die 37 Spalten hinaus, so scrollt der Bildschirminhalt nach links weg. Will man sich einen eingetippten Absatz noch einmal durchlesen, muß man mühsam hin und her springen. Die »Hilfe«, die das Programm hierzu bietet, kann



Bild 1. Die Übersicht kann in der Textverarbeitung schnell verlorengehen

Bild 2. 17 Spalten und 50 Zeilen stehen zur Verfügung

man nicht ganz ernst nehmen. Mit der F1-Taste kann man auf den linken, mit der F2-Taste auf den rechten Rand springen. Wie schon erwähnt, werden jeweils 37 Zeichen auf dem Bildschirm dargestellt. Zweimal 37 Zeichen, vom linken und vom rechten Rand aus gesehen, ergeben 74 Zeichen. Insgesamt sind in einer Zeile aber 77 Zeichen enthalten. Mit dieser Methode werden also immer die mittleren drei Zeichen abgeschnitten. Bleibt also doch nur das Scrollen über die Cursor-Tasten.

Das Zauberwort, um dieses Problem zu lösen, könnte »formatieren« lauten. 16 verschiedene Formatierungs-Befehle stehen dem Benutzer zur Verfügung. Allerdings nur zur Ausgabe auf dem Drucker. Auf dem Bildschirm sieht man davon nichts. Da man für die Druckerausgabe praktisch immer formatieren muß, hätte es für die Autoren dieser Software wohl keine Schwierigkeit bedeutet, den Text immer in 37 Zeichen pro Zeile auf dem Bildschirm darzustellen und beim Ausdrucken auf die gewünschte Zeilenbreite umzuformatieren.

Umständlich gestaltet sich auch das Einfügen und Löschen von Text. Zum Einfügen stehen dem Benutzer drei Möglichkeiten offen. Zuerst, wie auch im Programmiermodus, die INST/DEL-Taste, mit der man einzelne Zeichen einfügen und löschen kann. Benutzt man diese Funktion, so wird beim Einfügen der gesamte Text verschoben. Ist ein Absatz im Text, verschieben sich auch die Leerzeichen. Fügt man ein vierbuchstabiges Wort ein, sind die Textanfänge der folgenden Absätze jeweils um vier Zeichen eingerückt. Um beim Einfügen nicht den ganzen Text zu verschieben, steht der Befehl »SP« zur Verfügung. Die Kennzeichnung des Bereichs, der sich verschieben darf, erfolgt so: Der Cursor wird an das Ende des zu verschiebenden Bereichs gesetzt, zum Beispiel an das Ende des Absatzes, in dem das Wort eingefügt werden soll. Nun kennzeichnet man das Ende durch den »SP«-Befehl. Jetzt positioniert man den Cursor auf die Stelle, an der der Text eingefügt werden soll. Der Anfang braucht nicht extra gekennzeichnet zu werden. Hier drückt man so oft die INST/DEL-Taste, wie Buchstaben in dem einzufügenden Wort enthalten sind. Ist der Text eingefügt, steuert man den Cursor wieder auf die Ende-Position und löscht den Zeiger mit dem »EP«-Befehl. Fassen wir diese Aufgabenstellung noch einmal zusammen: In einem Brief soll in einem Absatz ein Wort eingefügt werden. Dabei soll sich der Text aber nur innerhalb dieses Absatzes verschieben. Dafür sind diese Befehle nötig:

◆ <Commodore> + C (um in den Befehlsmodus zu kommen
◆ **SP** <RETURN> (Ende-Zeiger setzen)
◆ Cursor auf den Anfang der Einfügung positionieren
◆ **INST/DEL** (Platz zum Einfügen schaffen)
◆ Text schreiben
◆ Cursor auf Ende-Zeiger positionieren
◆ <Commodore> + C (Befehlsmodus einschalten)
◆ **EP** <RETURN> (Ende-Zeiger löschen)

»Die Fähigkeit, Zeiger auf eine beliebige Stelle zu setzen, ist sehr nützlich und erleichtert das Einfügen eines neuen Textes oder das Löschen sehr.« heißt es im Handbuch. Mit den acht Aktionen, die man zum Einfügen durchführen muß, ist es wirklich »sehr« einfach. Diese Umständlichkeit läßt sich mit einem kleinen Trick umgehen. Setzen Sie bei jedem Absatzende mit der »SP«-Anweisung einen Ende-Zeiger. Jetzt können Sie an jeder beliebigen Stelle einfügen und müssen nur einmal, wenn der Text endgültig fertig ist, die Zeiger löschen.

Das Einfügen ganzer Zeilen ist mit dem »IL«-Befehl zu steuern und gestaltet sich ganz ähnlich.

Die guten Eigenschaften der Textverarbeitungssoftware gehen gegenüber den Nachteilen und den Umständlichkeiten fast ganz unter. Befehle, um blockweise zu kopieren und einzelne Textteile zu verbinden



Bild 4. Neben dem Netzschalter ist ein Resetknopf angebracht



Bild 5. Auf der Rückseite [...] User-Port, den veränderten [...]

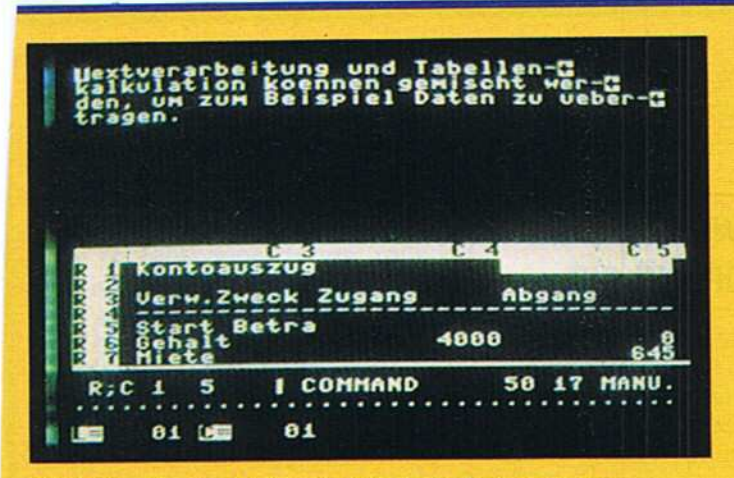


Bild 3. Eingeblendete Tabellenkalkulation in der Textverarbeitung

zeigen zwar einen guten Ansatz, werten die Software aber nicht auf.

Die Tabellenkalkulation ruft man aus dem Textverarbeitungsprogramm mit dem Befehl »TC« auf (Bild 2).

Plus 2 — Die Tabellenkalkulation

Ein wenig dürftig ist auch dieses Programm. Eintragungen in die Tabelle sind nicht ganz so kompliziert wie die Handhabung der Textverarbeitung, aber unter dem Begriff »benutzerfreundlich« ist auch dieses Programm nicht einzuordnen. Oftmals müssen zwei oder drei Anweisungen gegeben werden, um eine einzige Aktion auszuführen. Weiterhin läßt das Tabellenkalkulationsprogramm nur Berechnungsformeln mit einer einzigen Klammerebene zu. Will man kompliziertere Formeln verwenden, müssen sie so umgestaltet werden, daß intern nur jeweils eine Klammerebene abgearbeitet wird.

Die Kapazität der Tabellenkalkulation dürfte dagegen mit 850 Feldern, in einer festen Aufteilung von 17 Spalten und 50 Zeilen, für viele Anwendungen ausreichend sein.

Alles im allem dürfte die Tabellenkalkulation den Heimanwender und eventuell auch Kleinstbetriebe zufriedenstellen.

Positiv ist von den fest eingebauten Programmen nur die Dateiverwaltung zu bewerten.

Aufgerufen wird dieses Programm aus der Textverarbeitung oder der Tabellenkalkulation mit dem Befehl »TF« (Bild 3).

Plus 3 — Die Dateiverwaltung

Es können 17 Datenfelder mit bis zu 32 Zeichen pro Datensatz angelegt werden. Maximal sind 999 Datensätze pro Datei möglich; dies hängt allerdings von der Anzahl und Länge der Felder ab. Sobald der Aufbau des Datensatzes vom Programm auf der Diskette eingetragen ist, wird dem Benutzer mitgeteilt, wieviele Datensätze in dieser Datei eingetragen werden können.

Das Anlegen der einzelnen Datensätze wird, wie auch bei den anderen Programmen, über den Befehlsmodus, der über <Commodore> + C zu erreichen ist, gesteuert. Etwas umständlich, aber hier gewöhnt man sich recht schnell an die wenigen Befehle, die hierzu nötig sind.

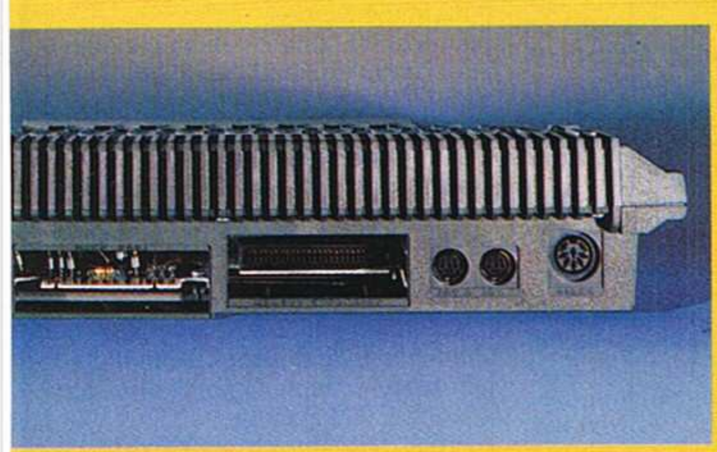
Man kann nach jeweils drei verschiedenen Oberbegriffen gleichzeitig sortieren lassen. In einer Adreßdatei zum Beispiel können so Listen erstellt werden, in denen erst nach dem Nachnamen sortiert wird, als zweites Kriterium nach dem Vornamen und als drittes noch nach Postleitzahlenbereichen.

Bemerkenswert ist hierbei, daß die Oberbegriffe, nach denen sortiert werden soll, frei wählbar sind. So können Listen aus demselben Datenbestand nach immer wieder unterschiedlichen Kriterien erstellt werden.

Auch die Suchbegriffe sind aus allen Oberbegriffen frei wählbar. Sucht man beispielsweise nach dem Namen »Eberhard«, spielt es keine Rolle, ob dies der Vor- oder Nachname ist. Alle Datensätze, in denen »Eberhard« vorkommt, werden ausgegeben. Lästig könnte es zwar sein, daß auch ein Datensatz mit einer »Eberhard-Krüger-Straße« mit ausgegeben wird, aber solche Gleichheiten dürften nur sehr selten vorkommen. Jeder Suchbegriff darf abgekürzt werden. Wissen Sie zum Beispiel nicht, ob sich »Eberhard« mit »d« oder »t« am Ende schreibt, so genügt die Angabe von »Eberhar«.

Plus 4 — Die grafische Auswertung

Bei dem Grafikprogramm haben die Programmierer der Software ihr »Meisterstück« vollbracht. Eine primitivere Form der Grafik ist kaum



len Netzanschluß, den seriellen Port, den veränderten Datasette-Anschluß, den sion-Port, die Joystickanschlüsse und den Videoanschluß.



Bild 6. Der HF-Ausgang ist an der Seite angebracht. Der zweite abgedeckte Ausgang ist in der amerikanischen Version ein Kanal-Umschalter.

noch zu realisieren. Betrachtet man die wirklich hervorragenden Grafikfähigkeiten des Plus/4 und sieht dann, was das Grafik-Programm aus den Zahlenkolonnen der Tabellenkalkulation macht, fühlt man sich auf den Arm genommen. Balken- und Liniengrafiken werden aus einzelnen »#« (Nummer-Zeichen) zusammengesetzt. Für die Ausgabe auf einem Drucker kann dies sinnvoll sein, da so die Grafiken auch mit einer Typenradschreibmaschine ausgegeben werden können. Doch zumindest für die Bildschirmausgabe hätte die Grafik mit den ganz normalen Basic V3.5-Befehlen leicht aufbereitet werden können.

Ein Plus

Einige besondere Aspekte der Plus/4-Software, die man nicht vernachlässigen sollte, sind bisher noch nicht erwähnt worden.

Sehr positiv ist zu bewerten, daß die Textverarbeitung mit der Tabellenkalkulation, der Dateiverwaltung und der grafischen Darstellung zusammenarbeitet, was bedeutet, daß Daten von einem Programm in das andere übertragen werden können. Hierzu gibt es mehrere Alternativen.

Über das eingebaute Windowing, mit dem man Teile des Bildschirms nutzen kann, ohne den anderen Bildschirmteil zu beeinflussen, kann man Teile der Tabellenkalkulation in die Textverarbeitung einblenden. So können Daten, die in Schriftstücken erforderlich sind, zum Beispiel in Geschäftsberichte übertragen werden.

Software: nur im Konzept gut

Benötigt man den gesamten Datenbestand einer Tabellenkalkulation für ein Dokument, läßt sich dies mit einem Kopierbefehl realisieren. Auch die grafische Auswertung wird auf diese Weise kopiert (Balken- oder Liniengrafik). Serienbrieferstellung kann in Zusammenarbeit mit der Dateiverwaltung realisiert werden.

Inwieweit die positiven Elemente der Plus/4-Software die negativen Aspekte aufwiegen, muß jeder Anwender für sich entscheiden. Die Idee der eingebauten Software ist sicherlich ein großer Schritt nach vorne auf dem Homecomputer-Markt. In der Ausführung hätten sich die Programmierer aber etwas mehr Mühe geben können.

Die Hardware überzeugt

Die Hardware des Plus/4 kann dagegen auf ganzer Linie überzeugen. Sehr gutes Basic 3.5 und 60 KByte freier Speicherplatz machen das Programmieren zum Vergnügen. Weiterhin stehen neben der guten Tastatur vier neben der SPACE-Taste angeordnete Cursor-Tasten und acht mit den gängigsten Befehlen belegte Funktionstasten zur Verfügung. Eine Version mit DIN-Tastatur wird zu einem Aufpreis von 100 Mark erhältlich sein. Neben dem Netzschalter ist ein Reset-Schalter angebracht (Bild 4), den man ja auch zum Abschalten der eingebauten Software braucht.

Ein Wermutstropfen sind die neuen Joystick- und Datasette-Anschlüsse (Bild 5). Sie sind nicht kompatibel zu denen des VC 20 und C 64. Umsteiger, die von diesen Computern kommen, müssen sich dieses Zubehör neu anschaffen. Im Gegensatz zum User-Port hat sich ebenfalls der Expansion-Port verändert. Also können auch Erweiterungs-Module und -Karten, die auf diesen Port ausgelegt sind, nicht mehr verwendet werden. Der HF-Ausgang wurde beim Plus/4 an die Seite verlegt (Bild 6).

Im Vergleich zum C 64 hat sich auch auf der Platine einiges geändert. So wurden der SID und der VIC, sowie Teile der VIA im TED integriert. Der neue 7501-Mikroprozessor ist kompatibel zu den bisher verwendeten 6502/6510 (Bild 7).

Schade, daß der Plus/4 nicht ohne eingebaute Software zu einem Preis zwischen 700 und 800 Mark zu haben ist, denn er bietet erheblich mehr als manch anderer Computer dieser Preisklasse. (rg)

Bild 7. Der SID und der VIC, sowie Teile der VIA sind nun im TED integriert. Der 7501-Mikroprozessor ist kompatibel zu 6502/6510. Alle Anschlüsse und strahlungsintensiven Teile sind abgeschirmt.

Auf einen Blick

CPU: 7501-Mikroprozessor/1.76 MHz kompatibel zu 6502/6510
RAM: 64 KByte
ROM: 32 KByte inklusive Betriebssystem und Basic V3.5 + 32 KByte für die eingebaute Textverarbeitung, Tabellenkalkulation, Dateiverwaltung und grafische Auswertung
Maße: Breite 305 mm, Tiefe 203 mm, Höhe 63 mm
Bildschirmdarstellung: 16 Farben (mit jeweils sieben Intensitätsabstufungen), 40 Zeichen in 25 Zeilen, Hires-Auflösung 320 mal 200 Punkte, Multi-Color-Auflösung 160 mal 200 Punkte
Preis: DIN-Tastatur (1398 Mark) und ASCII-Tastatur (1298 Mark)

Die Stimme des Meisters

Welch ungläubiges Staunen erweckte doch die erste Phonografentrommel seinerzeit. Kaum anders geht es dem Computerbesitzer heute, wenn er die eigene Stimme aus dem C 64 hört.



Die Hardware ist nur für die Aufnahme notwendig

Die Vielseitigkeit des Commodore 64 ist schon beinahe sprichwörtlich. Der Covox Voice-Master (siehe Bild) baut auf dieser Flexibilität auf. Er ist das erste Zusatzgerät für den C 64, mit dem es gelingt, die eigene Sprache elektronisch zu speichern und wieder abzurufen.

Im Speicher des Commodore werden die mit einem Mikrofon auf herkömmlichem Weg aufgenommenen Toninformationen als digitale Werte abgelegt. Ebenso wie alle Basic- oder Maschinenprogramme sind die Tondaten für das mitgelieferte Steuerprogramm jederzeit und vor allem sehr schnell verfügbar. Damit die aufgenommenen Tonimpulse aber auch wiedergegeben werden, bedient sich das Steuerprogramm des SIDs.

Töne mal ganz anders

Dieser produziert dann nicht, wie so oft, Musik oder die Geräuschkulisse für Spiele, sondern die Originalstimme des Benutzers. Ein verblüffender Effekt. Bisher bekannte Programme, wie etwa der SAM, mußten oft umständlich programmiert werden. Mit Voice-Master merkt sich der Commodore einfach die in ein Mikrofon gesprochenen Worte.

Auf diese Weise finden bis zu 64 einzelne Worte auf einmal im Speicher Platz. Da die Speicherung solcher Daten recht aufwendig und platzraubend ist, sind volle 64 Worte nur bei sehr kurzen Vokabeln möglich. In der Regel stehen etwa 40 programmierbare Worte zur Verfügung. Wer sich jetzt fragt, wieviel dann noch für sein Programm übrigbleibt, fragt richtig, denn umfangreiche Basic-Programme sind nicht mehr möglich. Dies gilt für die erste Version des Voice-Masters. Die ebenfalls mitgelieferte zweite Version läßt den gesamten Basic-Speicher unangetastet, hat aber nur noch etwa acht KByte für Tondaten (was nicht viel ist).

Einfachste Bedienung

Wer glaubt, daß ein solches Programm schwierig zu handhaben ist, wird freudig überrascht sein. Die Worte werden in einer FOR-NEXT-Schleife mit dem Befehl LEARN A eingegeben, die Variable A ist dabei die Nummer des Wortes. Zur Wiedergabe der Wörter (über den normalen Fernsehlautsprecher) genügt der Befehl SPEAK A. Ebenso simpel kann die Lautstärke (VOLUME B) oder die Sprechgeschwindigkeit (SPEED C) verändert werden. Wichtigstes Beeinflussungsmerkmal für die Töne ist die Aufnahmefrequenz (RATE D), die zwischen 5000 Baud und 12000 Baud einstellbar ist. Interessante und manchmal recht lustige Variationen der Worte sind das Resultat einer einfallsreichen Verkettung dieser Befehle. Mit etwas Geschick ist auch ein Echo-Effekt möglich.

Auf der Suche nach dem Zweck

Natürlich drängt sich bei einem solchen Zusatzgerät die Frage auf, wozu das Ganze eigentlich nützlich sein kann. Vordringlich fallen einem da Anwendungen wie die Sprachunterstützung bei Spielen (Achtung! Feind von rechts!), oder die akustische Unterstützung eines Auswahlmenüs ein. Daß auch wesentlich anspruchsvollere Anwendungen denkbar sind, zeigt das beigefügte Taschenrechnerprogramm, das jede eingegebene Zahl oder Rechenoperation mit Worten bestätigt. Mit einer auch beim Commodore 64 möglich erscheinenden Speichererweiterung von 256 KByte ist so eine akustische Eingabekontrolle jeder gedrückten Taste denkbar. Für Blinde eröffnet sich so die weite Welt der Microcomputer. Aber auch für Sehende ist das Erlernen der Programmiersprache Basic mit akustischer Unterstützung sicherlich einfacher.

Besonders angenehm ist es, daß man die eingegebenen Worte auf Diskette abspeichern kann. Im Bedarfsfall steht jeder beliebige Wortschatz innerhalb kürzester Zeit zur Verfügung. Zur Wiedergabe der Worte ist das Modul des Voice-Masters überflüssig, nicht aber die Software. Dieses kurze Programm muß in jedem Fall geladen beziehungsweise in ein Programm einbezogen werden. Programme mit eigener Sprachunterstützung stehen somit auch allen jenen zur Verfügung, die den Voice-Master nicht besitzen. Verändern können Sie den Wortschatz allerdings nicht mehr.

Sprachausgabe ohne Zusatz

Die Charakteristik jeder Stimme oder eines bestimmten Geräusches bleibt erstaunlich naturgetreu erhalten. Leider entspricht das mitgelieferte Mikrofon nicht dem Stand der Technik. Ein Versuch hingegen, Musik mit einem guten Mikrofon aufzunehmen, brachte ganz erstaunliche Ergebnisse. Der Commodore als Musikrecorder, ein mögliches, aber wohl etwas zu teures Verfahren. Für die reine Sprachein- und -ausgabe ist der Preis von 298 Mark gerade noch tragbar.

Zukunftsmusik

Wichtigste Erkenntnis ist aber die Enthüllung der ICs als Ton-Speichermedium der Zukunft. Vollkommen naturgetreue und beliebig beeinflußbare Reproduktion von Daten, seien es nun Tondaten oder Videoinformationen, sind das Ergebnis. Sollte sich die Preisentwicklung und die Leistungsfähigkeit der Speicherchips mit der gleichen Geschwindigkeit wie bisher weiterentwickeln, so wird es sicher bald Videokassetten geben, die kein Gramm Magnetband mehr enthalten. Der Voice-Master wird bis dahin aber sicher schon ein Urahn dieser Entwicklung sein.

(Arnd Wängler/aa)

Bezugsquelle: Microton, Computerprodukte, Bahnhofstr. 2, CH-2542 Pieterlen

16-KByte-Erweiterung umschaltbar

Im Gegensatz zur 8-KByte-Erweiterung ist die 16-KByte-Karte für den VC 20 auf einen Adreßbereich fest eingestellt. Mit wenig Bastelaufwand kann sie aber jedoch auch für den Steckmodulbereich genutzt werden.

Nicht wenige VC 20-Besitzer haben den Wunsch, ihre 16-KByte-Erweiterung VIC 1111 von Commodore auch für den Autostartbereich $A000-$BFFF (40960-49152), gelegentlich Modulbereich genannt, zu benutzen.

Der Autostartbereich bietet bekanntlich die Möglichkeit, unter gewissen Voraussetzungen sofort nach dem Einschalten an die Stelle $A000/$A001 zu springen und von dort natürlich an jede gewünschte Stelle im Programm, wobei noch besonders bemerkenswert ist, daß die Auslösung eines CPU-Resets mittels nachträglich eingebautem Taster, der beispielsweise Pin 6 mit Pin 2 an der Buchse des seriellen Ausgangs verbindet (»not connected« für Pin 6 in den diversen Büchern ist geschwindelt: Pin 6 liegt am System-Reset) dieselbe Wirkung wie das Einschalten hat: Es erfolgt ein Sprung in die Reset-Routine $FD22-$FD3C (64802-64828), Abfragen auf »A0CBM« in $A004-$A008 und gegebenenfalls Sprung indirekt über $A000/$A001 an die gewünschte Stelle. Bei Betätigung der Restore-Taste erfolgt ein Sprung indirekt über $A002/$A003.

Die immer wieder zu hörende Auskunft, die Erweiterung VIC 1111 sei nicht umschaltbar, stimmt nur bedingt. Man braucht lediglich einen geeigneten Schalter einzubauen. Es ist keine Elektronik nötig. Der Schalter hat mich 25 (fünfundzwanzig) Pfennige gekostet.

Die Erweiterung VIC 1111 ist mit durchkontaktierten Bohrungen versehen, in die man theoretisch zwei Vierfach-DIL-Schalter einbauen könnte. Dann wären die 16 KByte blockweise zu je 8 KByte überallhin (nach Block 1, 2, 3 oder 5) hin zu legen. Selbstverständlich macht man das nicht, da einerseits die Gefahr der Doppelbelegung und Zerstörung zu groß wäre und andererseits diese Stelle in der verschlossenen 16-KByte-Erweiterung ja nicht zugängig ist. Es wurde daher nur ein 8-KByte-Block umschaltbar gestaltet, und zwar zur wahlweisen Beschaltung von
[nichts/$4000-$5FFF],
[$2000-$3FFF/$4000-$5FFF],
[$6000-$7FFF/$4000-$5FFF],
oder [$A000-$8FFF/
$4000-$5FFF].



Bild 1. 16-KByte-Erweiterung mit eingebautem Umschalter

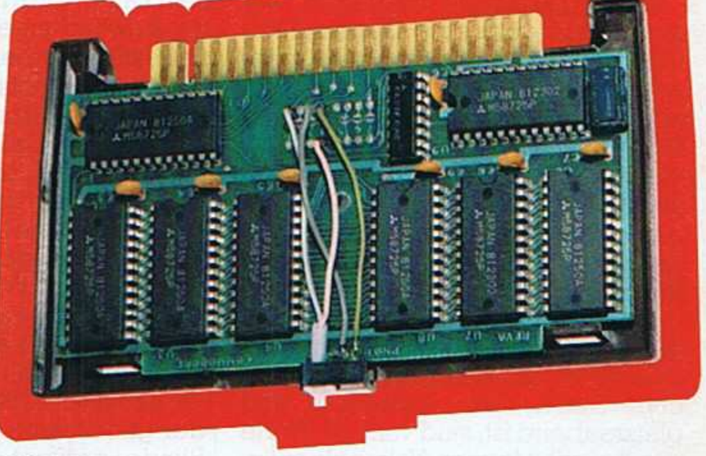


Bild 2. 16-KByte-Erweiterung mit eingebautem Umschalter, geöffnet.

Der Schalter wird mit Bohrmaschine, Feile und Lötkolben ohne großen Aufwand mechanisch recht sicher an der Stirnseite der »Cartridge« untergebracht (siehe Bild 1 und 2). Die elektrischen Anschlüsse sind Bild 3 zu entnehmen: Über die Anschlüsse 10, 11, 12 und 13 an der »Memory Expansion« werden der Erweiterung die voll decodierten Chip-Select-Signale $\overline{BLK1}$, $\overline{BLK2}$, $\overline{BLK3}$ und $\overline{BLK5}$ zugeführt. Über eine Lötbrücke geht das Signal $\overline{BLK1}$ beim uns interessierenden 8-KByte-Teil an den Zweifach-2-zu-4-Decoder 74LS139. Die Lötbrücke wird (mit etwas Entlötlitze ohne Schwierigkeiten) entfernt und die Schaltverbindungen, wie in Bild 3 gezeigt, so an den Vierfach-Umschalter herangeführt, daß keine Bereichsüberlappungen möglich sind.

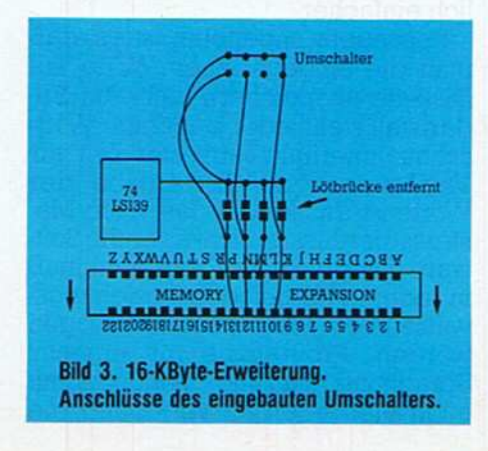


Bild 3. 16-KByte-Erweiterung. Anschlüsse des eingebauten Umschalters.

Zu bemerken wäre noch, daß es eine halbe Lebensaufgabe ist, die Ummantelung der 16-KByte-Erweiterung aufzubekommen: An den Schlitzen an der Stirnseite kann man mit einem Taschenmesser oder einem feinen Uhrmacherschraubenzieher gegendrücken und so die beiden Plastikschnäpper lösen, an der Steckerseite hilft rechts und links eine zurechtgebogene Büroklammer zum leichten herausziehen. Vorsicht, Bruchgefahr! Aber keine Angst: Die 16-KByte-Erweiterung funktioniert auch mit beschädigter Plastikhülle. Sie ist natürlich auch ohne jegliche Umhüllung hundertprozentig funktionstüchtig.

(Fred Behringer/ev)

Richtig verbunden — Video/Audio-Kabel für den C 64

Im Bedienungshandbuch zum C 64 ist auf Seite 142 die Belegung der Video/Audio-Buchse angegeben. Dieser Anschluß wird benötigt, wenn der C 64 nicht am Antenneneingang, sondern am Videoeingang eines Fernsehgeräts oder Monitors betrieben wird.

Neue Buchse beim C 64

Bei neueren Modellen des C 64 ist die im Handbuch beschriebene Video/Audio-Buchse durch eine andere ersetzt worden. Die Belegung der neuen Anschlußbuchse zeigt Bild 1. Die folgende Erläuterung der Signale gibt in knapper Form Aufschluß über deren Funktion.

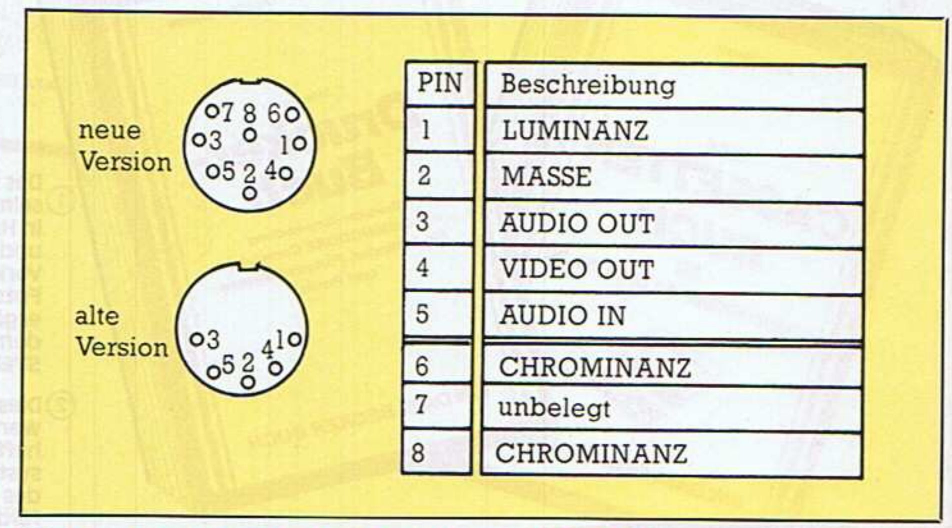
Die Bildsignale des C 64

Das Luminanzsignal, auch BAS-Signal genannt, ist ein Intensitätssignal. Es enthält die Helligkeitswerte zwischen Schwarz und Weiß und wird zur Ansteuerung eines monochromen Monitors oder SW-Fernsehgerätes benutzt.

Mit dem Chrominanzsignal (F-Signal) werden Farbinformationen übertragen. Das Chrominanz- und Luminanzsignal liefern zusammen die gesamte Farbbildinformation. Die getrennte Übertragung dieser Signale ist nicht allzu weit verbreitet. Nur wenige Monitore, wie der Commodore-Monitor 1701 oder 1702, haben getrennte Eingänge für Chrominanz und Luminanz. Getrennte Signale führen zu einer schärferen Bildwiedergabe als das FBAS-Signal.

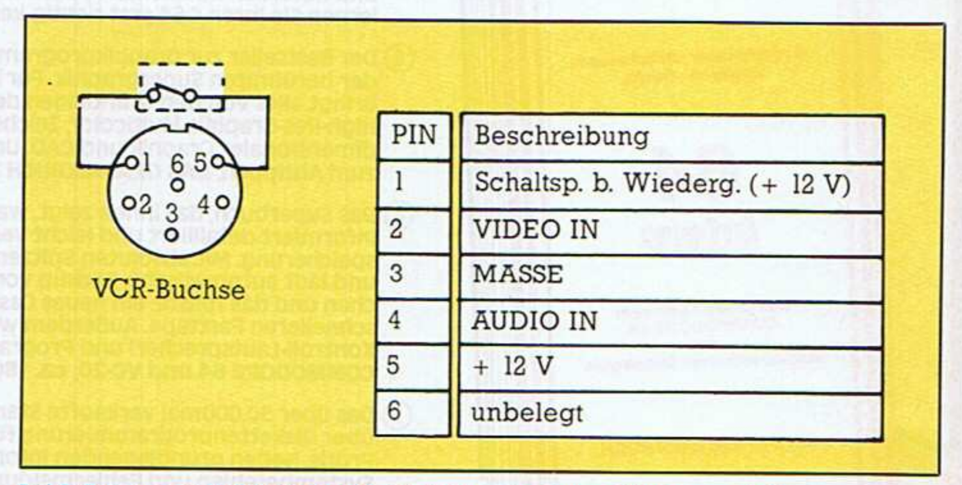
Das VIDEO OUT-Signal ist kompatibel zum FBAS- oder PAL-Signal und setzt sich aus dem Luminanz- und Chrominanzsignal zusammen. Es enthält alle Informationen, die der Aufbau eines Farbbildes erfordert. Dieses Signal erlaubt es, einen Farbfernseher oder Monitor mit PAL-Videoeingang an den C 64 anzuschließen, ohne den Umweg über den Antennenanschluß zu nehmen. Die erforderliche Modulation/Demodulation führt dabei nämlich zu einem Qualitätsverlust.

Die Beschaltung einer Videobuchse herkömmlicher Bauart



PIN	Beschreibung
1	LUMINANZ
2	MASSE
3	AUDIO OUT
4	VIDEO OUT
5	AUDIO IN
6	CHROMINANZ
7	unbelegt
8	CHROMINANZ

Bild 1. Beschaltung der alten und neuen Video/Audio-Buchse beim C 64



PIN	Beschreibung
1	Schaltsp. b. Wiederg. (+ 12 V)
2	VIDEO IN
3	MASSE
4	AUDIO IN
5	+ 12 V
6	unbelegt

Bild 2. Belegung der VCR-Buchse herkömmlicher Bauart mit dem Aktivierungsschalter

SCART-Buchse

PIN	Beschreibung (auszugsweise)
2	AUDIO IN (Rechter Kanal)
4	MASSE
6	AUDIO IN (Linker Kanal)
18	Schaltspannung 1 (+ 12 V)
16	Schaltspannung 2 (+ 1...3 V)
17	MASSE
20	VIDEO IN

Bild 3. SCART-Buchse. Belegung der Pins, die zum Anschluß des C 64 benötigt werden

Sie müssen kein Elektronikprofi sein oder Elektrotechnik studiert haben, um einen Fernseher mit Videoeingang oder einen Monitor an den C 64 anzuschließen.

Wenn Sie wissen, an welchem Ende ein Lötkolben heiß wird, dann dürfte diese Bauanleitung für Sie ein klarer Fall sein.

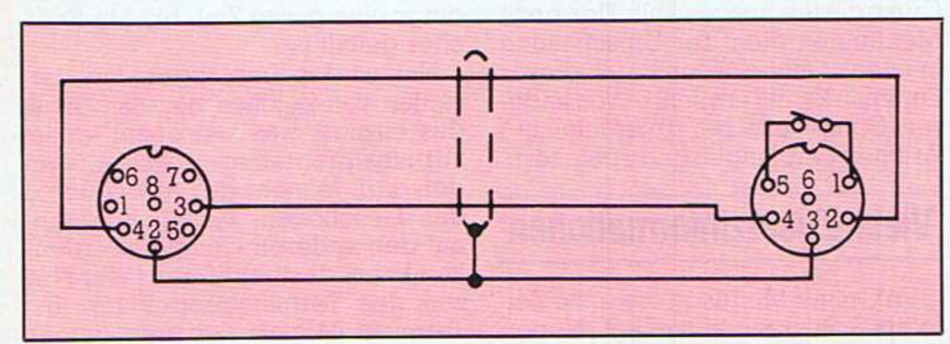


Bild 4. Anschluß eines SW-Fernsehgerätes oder eines monochromen Monitors

(VCR) können Sie Bild 2 entnehmen. Bild 3 zeigt die der neuen, genormten SCART-Buchse.

Über das AUDIO OUT-Signal wird der Ton des C 64 übertragen. Hat der Monitor kein Tonteil, hilft nur eine Nachrüstung oder ein externer Verstärker.

Der Vollständigkeit halber sollte der AUDIO IN-Eingang nicht unerwähnt bleiben. Über diesen werden dem Soundchip SID 6581 des C 64 externe Audiosignale zugeführt, die sich bei entsprechender Programmierung be- und verarbeiten lassen. Da dieser Eingang ungepuffert ist (nur über einen Elko 10µF/25V entkoppelt) und direkt an das SID führt, ist höchste Vorsicht geboten. Das Handbuch sagt nichts über die zulässigen Grenzwerte aus. Vermutlich ist der TTL-Pegel (+5V) das äußerste Limit.

Die Verbindung

Um den Computer und den Monitor beziehungsweise das Fernsehgerät miteinander zu verbinden, benötigt man ein Verbindungskabel, das leicht aus einem abgeschirmten zweiadrigen Kabel und den passenden Steckern angefertigt wird. Die Teile gibt es im Elektronikfachhandel. Bild 4 zeigt ein Kabel für die Verbindung C 64 — monochromer Monitor/SW-Fernsehgerät. Die Bilder 5 und 6 zeigen den Anschluß an einen Farbmonitor/Farbfernsehgerät mit herkömmlicher VCR-Buchse und dem neuen SCART-Anschluß.

Bei Fernsehgeräten muß zur Umschaltung auf die VCR-Buchse Pin 1 auf eine Schaltspannung von +12V gelegt werden. Dies läßt sich mit einem Schalter über Pin 1 und 5 des VCR-Steckers realisieren. Wer auf den Schalter verzichten möchte, lötet zwischen die beiden Pins einfach eine Drahtbrücke. Soll der Fernseher dann wieder »normales« Programm empfangen, muß der Stecker gezogen werden.

Beim SCART-Stecker ist die Umschaltung nicht ganz so einfach zu verwirklichen, da keine Schaltspannung zur Verfügung steht. Die benötigte Spannung von +1 bis 3V kann allerdings am C 64 abgegriffen werden: am Pin 2 des User-Ports liegt die Spannung +5V/100mA an. Beim VC 20 wird der Pin 1 der VIDEO/AUDIO-Buchse genommen. Über einen Widerstand von 330 Ohm legt man die Spannung an Pin 16 des SCART-Steckers. Dies hat den Vorteil, daß beim Einschalten des Computers eine automatische Umschaltung des Fernsehgerätes von »Fernseher« auf »Datensichtgerät« erfolgt.

(Dipl.-Ing. R. Kurzhals/hm)

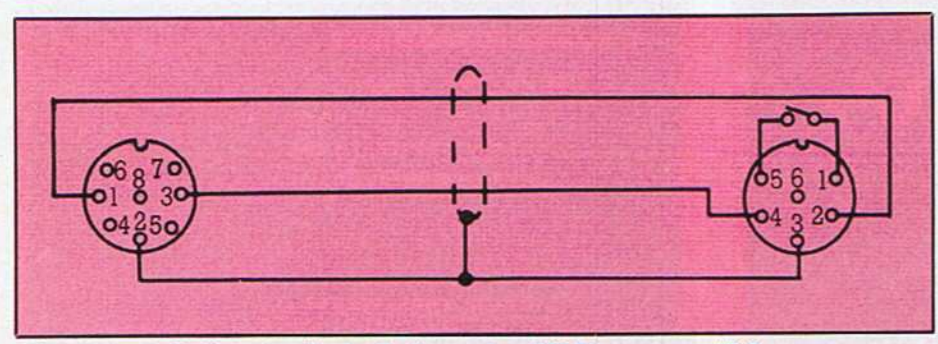


Bild 5. So ist ein Farbfernsehgerät oder Monitor an einen VCR-Eingang anzuschließen

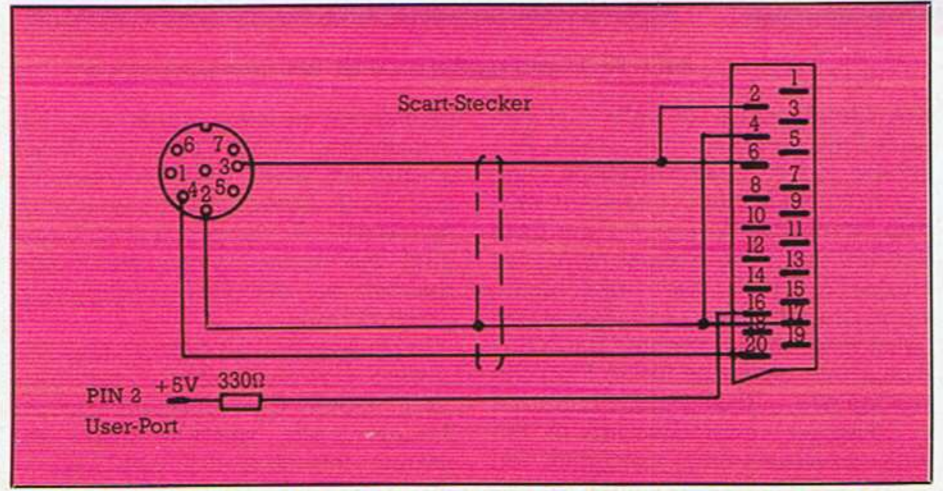


Bild 6. Die SCART-Verbindung. Am User-Port, Pin 2, können 5V abgegriffen werden, um zwischen Computer und Tuner umschalten zu können.

Literatur: Funkschau 8/1983, Seite 84-85.
MC 11/1983, Seite 50. Das große Werkbuch Elektronik, Nührmann, Franzis Verlag.

Terminal 64 — Schwer auf Draht

Die Datenfernübertragung mit dem Commodore 64 wird immer interessanter. Seit nun auch gute Akustikkoppler mit FTZ-Nummer zu günstigen Preisen erhältlich sind, gehört eine DFÜ-Ausrüstung fast schon zur Grundausstattung eines Computers.

Es ist kaum ein halbes Jahr her, als der C 64-Besitzer, ausgestattet mit einer RS232-Schnittstelle (noch vom VC 20), einem zusammengelöteten Akustikkoppler und einem kleinen, unkomfortablen DFÜ-Programm noch als Exot unter den Benutzern der verschiedenen Mailboxen galt. Damals war dies die Welt der Epsons, DECs und IBMs, heute ist das eher umgekehrt. Einen wesentlichen Beitrag zu dieser Entwicklung leisteten findige Programmierer, die dem Anwender immer komfortablere DFÜ-Programme zur Verfügung gestellt haben.

Gute Software ist wichtig

Jede auch noch so teure Geräteausstattung bleibt ohne Wert, solange nicht die entsprechende Software erhältlich ist. Mit Terminal 64 wird nun ein Programm angeboten, das den Anspruch erhebt, komfortabel und leistungsfähig zu sein. Wir haben es im Praxisbetrieb getestet. Angeschlossen waren neben der obligatorischen RS232-Schnittstelle und dem Verbindungskabel ein Minimodem 3005.

Nach dem Laden steht dem Anwender das erste der verschiedenen Untermenüs zur Verfügung (Bild 1). Im Gegensatz zur sonst üblichen Praxis werden die Unterfunktionen nicht über Zahlen, sondern mit den Cursortasten ausgewählt. Der erste Menüpunkt dient zur Umschaltung in den Online-Modus. In dieser Funktion kann Terminal 64 von anderen Computern per Telefonleitung Zeichen empfangen und speichern.

Viele Zusatzinformationen

Im Online-Modus ist die erste Zeile als Statuszeile ausgelegt. Neben dem Schriftzug »Online« dient die Statuszeile der Übermittlung diverser Informationen. Es handelt sich dabei um sogenannte »Statusflags«. Für alle, die ihre Telefonrechnung selber bezahlen müssen, ist die hier angezeigte Zeitinformation sicherlich sehr kostensparend, denn beim Stöbern in Datenbanken ist Zeit nur ein sehr relativer Faktor. Das zweite Flag (snd) gibt Auskunft darüber, ob der Commodore gerade aus seinem Puffer sendet. Nach dem Abschluß jedes Sendevorgangs kann am dritten Flag (end) erkannt werden, wann der Computer mit dem Senden fertig ist. Ebenso wird das vierte Flag (sto) revers geschaltet, wenn empfangene Zeichen gespeichert werden. Am rechten Rand der Statuszeile zeigt das fünfte Flag die verfügbare Restspeicherkapazität in Zeilen zu vierzig Zeichen an. Bei einer Gesamtspeicherkapazität von zirka 40000 Zeichen dauert es allerdings eine ganze Zeit, bis der Speicher gefüllt ist.

Nun ist das Empfangen und Darstellen von Zeichen natürlich nicht das einzige, was von einem guten Terminalprogramm erwartet werden darf. Je mehr der Anwender in das Geschehen, beziehungsweise auf den anderen Computer einwirken kann, desto größer ist der Nutzen des Terminalprogramms. Bei Terminal 64 wird der empfangene Text sowohl auf dem Bildschirm dargestellt, als auch in den Computer geladen. Dabei können Speicherbeginn und -ende entweder vom Benutzer selbst oder auch vom Hostrechner gesteuert werden. Beim benutzergesteuerten Speicher wird mit der F1-Taste bestimmt, welche Textpassagen gespeichert werden sollen. Unwichtige Teile können auf diese Weise per Tastendruck ausgeblendet werden, damit der Pufferspeicher nicht unnötig vollgeschrieben wird. Oftmals kann es aber auch vorteilhaft sein, wenn der sendende Computer Speicherbeginn und -ende steuern kann, zum Beispiel bei der Übertragung von Programmen.

In diesem Fall ist der Job des Benutzers von Terminal 64 recht einfach, denn die entsprechenden Steuerzeichen werden vom Programm unterstützt und gegebenen-

Bild 1. Das Hauptmenü von Terminal 64

Bild 2. Die Diskettenoperationen mit den Umwandlungsmöglichkeiten

falls ausgeführt. Der umgekehrte Fall, in dem der Commodore zum sendenden Computer wird, ist natürlich auch denkbar. Mit CTRL B beziehungsweise CTRL C wird nun der Speicherbeginn und das Speicherende des empfangenden Computers gesteuert.

Das Senden von Zeichen kann mit Terminal 64 grundsätzlich auf drei verschiedenen Wegen realisiert werden. Zur Verfügung stehen entweder die Tastatur, die mit beliebigen Strings belegbaren Funktionstasten oder der Pufferspeicher. Die erste und zweite Methode dienen in der Regel dem direkten Dialog mit anderen Computern, bei denen sich Sender und Empfänger abwechseln. Das Senden aus dem Pufferspeicher lohnt sich meist dann, wenn längere Texte, beispielsweise Listings, übertragen werden sollen. Dabei besteht beim Übertragen des Pufferspeichers die Wahlmöglichkeit zwischen zeilenweisem oder kontinuierlichem Senden. Auch die für manche Hostcomputer gelegentlich notwendige Verzögerung der Datenübertragung, kann mit Terminal 64 in drei verschiedenen Stufen eingestellt werden.

Eines der interessantesten Anwendungsgebiete für den Modembetrieb ist die Übertragung von Programmfiles. Dies geht im Prinzip nicht viel anders vor sich, als die Übermittlung von Texten oder sonstigen Mitteilungen. Da im Modembetrieb aber eine Übermittlung gemäß dem ASCII-Standard notwendig ist, muß der Programmcode vor dem Senden in ein ASCII-File umgewandelt werden. Das geschieht vor dem Laden von Terminal 64 am einfachsten mit dem LIST-Befehl: 1. Programm laden, dann im Direktmodus OPEN 2,8,2, "FILENAME, S, W": CMD2 : LIST: CLOSE2 eingeben. Dabei ist zu beachten, daß im Programm keine Commodore-spezifischen Steuerzeichen enthalten sein dürfen (durch CHR$-Codes ersetzen). Ein so präpariertes Programm kann nun als sequentielles File von Terminal 64 in den Pufferspeicher geladen und dann gesendet werden. Für den Empfang von im Quellcode vorliegenden Programmen stellt Terminal 64 sogar eine eigene Umwandlungsroutine bereit. Für ein empfangenes und im Pufferspeicher abgelegtes Programm wird im Untermenü Diskettenoperationen der Punkt Fileumwandlung angewählt. Nach dem Eingeben des Filenamens wird das transformierte File als Basic-Programm auf Diskette geschrieben. Der Pufferspeicher kann, ebenfalls vom Diskettenmenü aus, aber auch ohne Umwandlung zur späteren Weiterverarbeitung abgespeichert werden (Bild 2).

Mit allen seinen komfortablen Eingabefunktionen und seiner kompletten Ausstattung, sogar mit einem Standard ASCII- und einem deutschen Zeichensatz, ist der Terminal 64-Editor beinahe schon ein kleines Textverarbeitungsprogramm (Bild 3). Da auch alle Funktionen zum Speichern und Laden von Texten vorhanden sind, lohnt es manchmal, bestimmte Texte, die später gesendet werden sollen, gleich in den Pufferspeicher von Terminal 64 zu schreiben. Damit aber nicht genug, denn Terminal 64 kann alle sequentiellen Files lesen. Textverarbeitungsprogramme können zur Erstellung eines Briefes herangezogen werden.



Bild 3. Der Editor des Pufferspeichers

parameter ändern
Baudrate: 150 300 600 1200 2400
Datenbits: 8 7 6 5
Stopbits: 1 2
übertragung: Halbduplex Vollduplex
Parität: gerade ungerade keine
achtes Bit: keins null eins
Delay: aus 1 2 3
Auto-Return: aus 40 64 80 200
Tastatur: deutsch ascii
Drucker: cbm ascii

Bild 4. Alle Einstellparameter von Terminal 64

Flexibilität großgeschrieben

Wichtigstes Kriterium eines Terminalprogramms ist natürlich seine Flexibilität. Wer nur deshalb, weil Einstellparameter fehlen, die Hälfte aller verfügbaren Datenbanken erreicht, wird sicher bald enttäuscht sein. Terminal 64 ist in dieser Hinsicht vorbildlich. Das umfangreiche Parametermenü (Bild 4) läßt fast keine Wünsche offen. Baudraten von 150 bis 2400, Anzahl der Datenbits, Stopp-Bits, Halbduplex/Vollduplex und vieles mehr, stehen zur Wahl. Besonders interessant sind natürlich die beiden letzten Punkte, denn sie verwandeln den Commodore in einen Computer mit vollständiger DIN-Tastatur. Da Terminal 64 auch viele am Commodore anschließbaren Drucker unterstützt, entfallen im Zusammenspiel mit einem Umlautfähigen Drucker unliebsame Falschzeichen auf dem Ausdruck. Aber auch der ASCII-Zeichensatz entspricht der Norm. Die dadurch erreichte Kompatibilität mit anderen Hostcomputern kann mit Recht als fast hundertprozentig bezeichnet werden.

Datensicherheit, Leistungsfähigkeit und Komfort sind wohl die treffendsten Attribute für dieses Terminalprogramm. Bis auf die fehlende Darstellung von 80 Zeichen auf dem Bildschirm, kann Terminal 64 als Wegbereiter und neuer Standard für Terminalsoftware für den Commodore 64 gelten. Durch den großen verbleibenden Pufferspeicher ist mit diesem Programm ein effizientes Arbeiten in der Datenfernübertragung möglich.

(Arnd Wängler/rg)

Bezugsquelle: Electronic Universe, Hindenburgstr. 98, 2120 Lüneburg. Preis inklusive Schnittstelle 169 Mark.



Bild 1. »Löwe Leopold« ist immer dabei: Westermann Software.

Nachhilfe auf Knopfdruck

Können in Zukunft Mathematiklernprogramme den bewährten Nachhilfelehrer ersetzen? Um diese Frage zu klären, testeten wir zwölf Programme von fünf Anbietern, die den Mathematikunterricht in den unteren Klassen unterstützen sollen.

Ein Mathematikprogramm sollte danach bewertet werden, wieviel Wissen, in bezug zum jeweiligen Klassenlehrplan, vermittelt wird. Dabei spielt die Aufmachung eine große Rolle. Die Programme sollten auf psychologische und pädagogische Erkenntnisse und Grundsätze aufbauen: keine Bestrafung bei falschen, aber ein Lob bei richtigen Lösungen. Ein falscher Tastendruck darf nicht zu irritierenden Fehlermeldungen oder Systemabstürzen führen.

Lernen mit Leopold

Der Schulbuchverlag Westermann reiht sich mit sieben Programmen in die Avantgarde der Lernsoftware ein. Wir haben zwei der drei von Westermann angebotenen Mathematikprogramme getestet:
Bruchrechnung Teil 1 und
Prozentrechung.

Zu Beginn des »Unterrichts« wählt der Schüler die Art und Anzahl der Aufgaben. So kann man zum Beispiel bei Prozentrechnung zwischen dem Grund- oder Prozentwert und dem Prozentsatz entscheiden. Die Aufgaben werden in zwei Schritten, wie im Schulheft, ausgerechnet. Im ersten Lösungsschritt wird das Lösungsschema verlangt, im zweiten das Ergebnis. Stimmt die Lösung, freut sich das Lektionsmaskottchen »Löwe Leopold« und zeigt kleine »Dressurakte«. Ist das Ergebnis falsch, röhrt Leopold und fragt, ob man noch einen Lösungsversuch unternehmen will. Wenn nicht, zeigt das Programm den Lösungsweg und das Ergebnis. Auf Wunsch kann man sich mit F1 das Lösungsschema anhand eines Beispiels ausführlich zeigen lassen.

Die Mathematikprogramme von Westermann sind grafisch ansprechend aufgebaut. Das Lob dürfte Kinder zum Rechnen weiterer Aufgaben animieren. Bei falschen Lösungen wird der »Schüler« von seinem elektronischen Lehrer nicht mit frustrierenden Kommentaren oder Noten traktiert, die mehr zum Ausschalten des Computers als zum Lernen anspornen.

Für 69 Mark (nur auf Diskette) pro Lektion, bekommt man also Programme von Westermann, die zur Übung von einfachen Rechenaufga-



Bild 2. So präsentiert Hagemann Rechenaufgaben.



Bild 3. »Junior Mathemat« will erst den Namen des Schülers wissen.

ben geeignet sind, wobei der Kopf schon mal raucht, wenn alle Aufgaben ohne Taschenrechnner erledigt werden.

Lernen — Trainieren — Begreifen

Ein anderer Lehrmittelverlag versucht sich ebenfalls mit Lernprogrammen auf dem Markt zu behaupten: Hagemann aus Düsseldorf. Hagemann bietet verschiedene Mathelektionen an, die sich auf sieben Kassetten und sechs Disketten verteilen. Die Kassetten kosten ab 38,50 Mark, die Disketten zwischen 65 und 69 Mark.

Am besten gefiel, wie die Aufgaben zu lösen waren. Bei den vier Grundrechnungsarten erfolgen die Rechnungen wie auf einem Blatt Papier. Überhaupt nicht begeistern konnte die Art der Eingaben. Bei den Grundrechungsarten wird durch eine GET-Routine eingelesen, was Tippfehler sofort als Rechenfehler abstempelt. Bei Prozent- und Zinsrechnung läuft die Eingabe über einfache INPUTs, was dazu führt, daß das Programm unsinnige Eingaben erlaubt und der Computer dann Fehlermeldungen des Betriebssystems ausgibt. Wie Hagemann versicherte, sollen ab Dezember neue Programmversionen auf den Markt kommen, die Eingabefehler abfangen. Die optische Aufmachung der Lernprogramme für Grundrechenarten ist für die Zielgruppe der Rechenanfänger angemessen, könnte aber besser gestaltet sein. Dafür gibt es ausführliche Beispiele mit Erklärungen bei falschen Antworten. Besonders ist hervorzuheben, daß die Programme über keinen Kopierschutz verfügen. Der Schüler kann das Programm listen und anschauen, um einen Bezug zum Computer zu bekommen. Das soll auch bei den neuen Versionen so bleiben, falls es programmtechnisch möglich ist.

Mathe für den Junior

Mit »junior mathemat« bietet Data Becker ein Spiel- und Lernprogramm an. Es soll, wie im Vorwort zum Programm steht, den Mathematikunterricht, von der Grundschule an, begleiten. Das Programm umfaßt eine Vielzahl von Aufgabenstellungen. Neben den vier Grundrechenarten soll »junior mathemat« auch beim Üben von Mengenlehre, Maßeinheiten, Zahlenbetrachtungen und Ungleichungen zur Seite stehen.

Bevor man sich ans Werk macht, muß erst langwierig für jeden Schüler eine Datei auf einer eigenen Diskette kreiert werden. Diese enthält Angaben über die Anzahl und Art der Aufgaben, die gestellt werden sollen, und den Namen des Schülers. Beim Formatieren dieser Diskette ist es dringendst zu empfehlen »junior mathemat,ma« als Diskettennamen einzugeben, um nicht beim späteren Arbeiten einen »Diskettenfehler« zu erzeugen. Wegen des vorhandenen Kopierschutzes kann die Datei mit den persönlichen Angaben leider nicht auf der Programmdiskette abgespeichert werden und macht dadurch den Schüler zum Diskjockey.

Hat man wieder die Programmdiskette eingelegt und »junior mathemat« gestartet, wird der Name abgefragt. Entspricht die Schreibweise nicht exakt der in der Datei, wird erklärt, daß der Name falsch eingegeben ist. Man erhält keinen Hinweis, wo der Fehler zu suchen ist.

Pädagogisch nicht sinnvoll ist die Art, mit der ein Schüler bei einem falschen Ergebnis konfrontiert wird: Pro falsche Antwort sind drei Fragen mehr zu beantworten. Diese Strafarbeit läßt das Programm schnell unattraktiv werden. Als Belohnung bietet das Programm ein grafisch unbedeutendes Fangspiel an, das nicht unbedingt zum Weiterüben reizt.

Hat der Schüler den Aufgabenkatalog durchgearbeitet, steht eine Erfolgskontrolle zur Verfügung, die nach jeder Übungsstunde aktualisiert wird. Wegen der frustrierenden Art, mit der dem Schüler, bei falschen Antworten, Zusatzaufgaben aufgebrummt werden und der schwachen Benutzerführung konnte »mathemat junior« nicht überzeugen. Das Programm kostet 69 Mark (nur Diskette erhältlich).

Sucess With Math — Lernen mit Erfolg

Schon seit längerem bietet CBS-Software Lernprogramme auf dem amerikanischen Markt an. CBS will nun den deutschen Markt mit einer Serie von Mathematikprogrammen erobern.

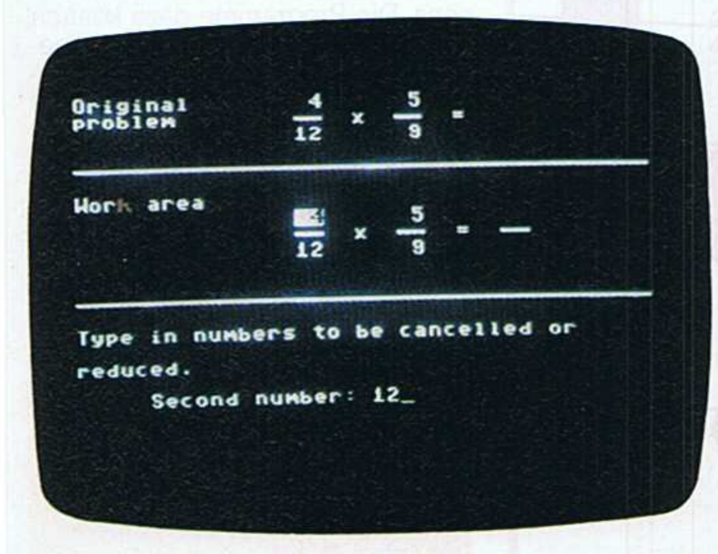


Bild 4. »Success With Math« von CBS.

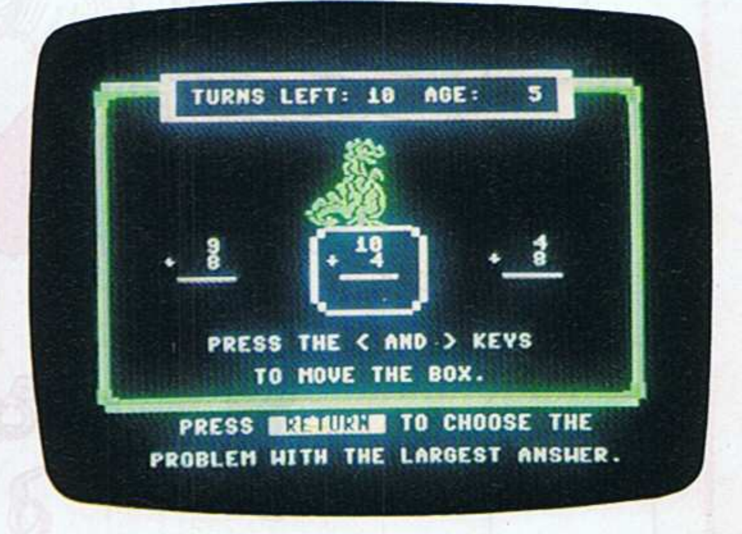


Bild 5. »Digitosaurus« überprüft das Ergebnis. Hes.



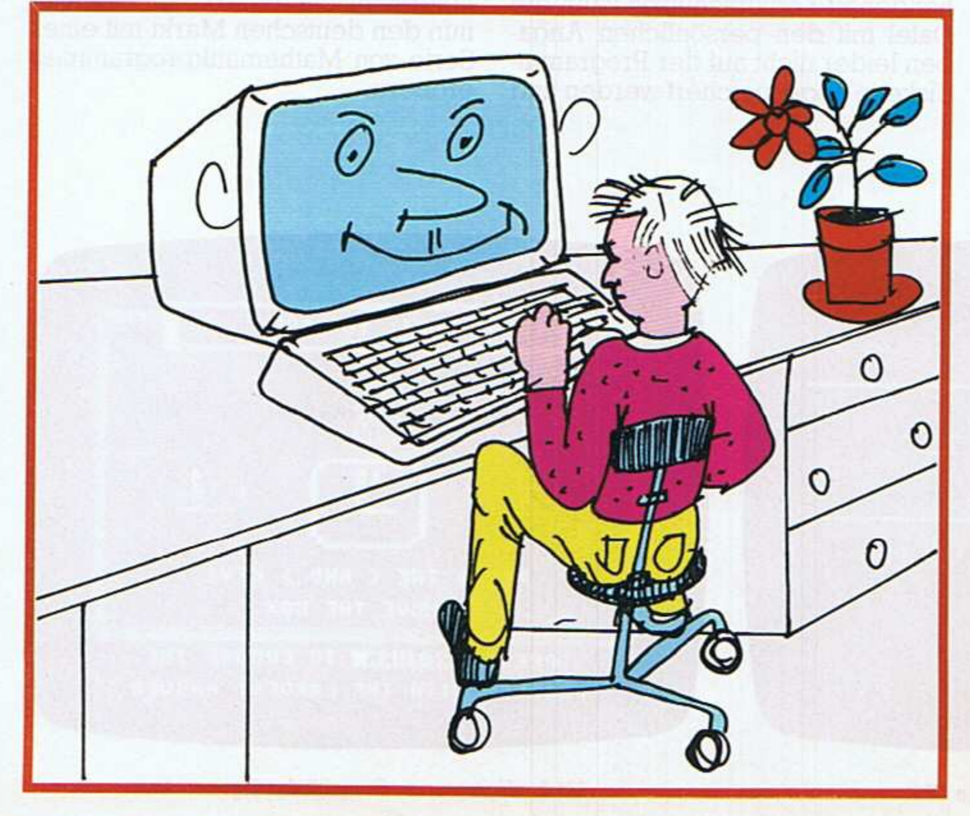
Bild 6. Die »Math Mansion« von Hes.

»Sucess With Math« heißt der Slogan und die Programmserie. Das Aufgabenspektrum der Serie reicht von den Grundrechungsarten bis zu linearen und quadratischen Gleichungen. Das erste der acht Lernpakete umfaßt die Rechenarten Addition und Subtraktion. Die einzelnen Übungen werden dem Schüler lieblos vorgesetzt, so daß der Spaß am Lernen schnell schwindet. Bei bestandenen Lektionen winkt kein Lob in Form von Grafik oder Sound.

Statt dessen erhält man eine trockene Fehleranalyse serviert. In jedem Schulbuch wird der Lernstoff attraktiver angeboten.

Die anderen Lernpakete wie Multiplikation und Division konnten ebensowenig überzeugen, wie die letzten beiden Lektionen über lineare und quadratische Gleichungen.

Ein zusätzlicher entscheidender Nachteil der CBS-Lernpakete ist die Tatsache, daß die Programme in Englisch abgefaßt sind und dadurch die größte Zielgruppe der Anwender, Schulkinder in den unteren Klassen, völlig verfehlen. Von Programmen dieser Preisklasse (Kassette 49 Mark, Diskette 69 Mark) sollte man mehr erwarten können.



Im Labyrinth der Mathematik

Mit »TRI-MATH« bietet Hesware ein Lernprogramm an, das schon nach dem Laden durch die hervorragende grafische Aufmachung die Aufmerksamkeit des Schülers auf sich lenkt. Das Programm besteht aus drei Teilen: zwei Science-fiction-Teilen und einem Adventure. In jedem der Programmteile soll der Schüler, auf recht spielerische Weise, Kopfrechnen üben. Die Aufgaben entsprechen dabei den Anforderungen der unteren Schulklassen. Es werden die Grundrechenarten verlangt. Im ersten Spiel gilt es schneller zu rechnen als ein kosmischer Gegner. Im zweiten müssen für einen »Digitosaurus« Aufgaben gelöst werden, damit dieser alt und weise wird und schließlich mit seinem Helfer in der »Hall of Fame« landet. Der dritte Teil, ein Adventure, bei dem richtige Lösungen Tür und Tor öffen, besticht durch seine grafischen Fähigkeiten.

Trotz der guten Eigenschaften wird sich »TRI-MATH« schwer am deutschen Markt durchsetzen, da das Programm in Englisch abgefaßt ist und nur einfache Aufgaben stellt.

Es war nicht immer so

Mathematiknachhilfe mit dem Computer ist eine neue Art des Lernens. Die Programme dazu können ohne weiteres als Prototypen bezeichnet werden, die noch verbesserungsbedürftig sind. Man sollte das Programmangebot gründlich prüfen, da, neben einigen brauchbaren Programmen, noch sehr viele existieren, die den Namen »Lernprogramm« nicht verdienen. (hm)

Info:
Lernen mit Leopold
Westermann Software, Postfach 5520, 3300 Braunschweig
Lernen — Trainieren — Begreifen
Hagemann, Karlstr. 20, 4000 Düsseldorf
Mathe für den Junior
Data Becker, Merowingerstr. 30, 4000 Düsseldorf
Sucess With Math
CBS-International, Rue Chateau 1, F-92200 Neuville
Im Labyrinth der Mathematik
Hesware, Human Engineering Software, 150 North Hill Drive, Brisbane, CA 94005

Assembler im Test

Teil 2

Bevor wir auf die vier diesmal getesteten Assembler zu sprechen kommen, vorher noch ein paar Bemerkungen über die Zielgruppe dieser Programm-Pakete.

Die in der letzten Ausgabe vorgestellten Assembler kann man sowohl vom Preis als auch von den Möglichkeiten her zu den professionellen Programmentwicklungssystemen zählen (vielleicht mit Ausnahme des T.EX.AS., der als Lehrsystem bezeichnet wird). In ihnen sind Funktionen enthalten, die selbst fortgeschrittene (Assembler-)Programmierer erst nach relativ langer Zeit und nach andauernder Benutzung beherrschen und anwenden. Dann allerdings möchte man auf den gebotenen Komfort und die mächtigen Befehle (zum Beispiel Makro-Bibliotheken, bedingte Assemblierung, etc.) nicht mehr verzichten.

Assembler, wie sie in dieser Ausgabe vorgestellt werden, wenden sich mehr an den Assembler-Anfänger und Gelegenheitsprogrammierer. Das heißt jedoch nicht, daß mit ihnen nicht auch vernünftig gearbeitet werden könnte. Für kleinere Projekte (klein bezieht sich nicht nur auf den Umfang, sondern auch auf die Komplexität) reichen diese Assembler nicht nur aus, sondern erfüllen durchaus ihren Zweck. Da es keine unnötigen Funktionen gibt, braucht man sich nicht mit einem Ballast von Kommandos herumzuschlagen. Die Geschwindigkeit ist (abgesehen vielleicht vom Mastercode-Assembler) völlig ausreichend. Mit Ausnahme vom Profimat-System sind in den anderen drei Programmen Editor, Monitor und Assembler zusammengefaßt, so daß Maschinensprache-Programme in einem Arbeitsgang geschrieben, assembliert und getestet werden können. Auch das ist ein Vorteil. Nun ist es aber nicht so, daß ein Assembler unbedingt notwendig zum Programmieren in Maschinensprache ist. Ein guter Monitor, wie zum Beispiel der SMON, stellt fast alle Hilfsmittel zur Verfügung (sehr wichtig zum Beispiel: Label). So wurde unter anderem das Programm HYPRA-LOAD aus Ausgabe 10/84 in monatelanger Arbeit, aber lediglich mit einem Monitor erstellt. Ein Assembler hat jedoch den Vorteil, daß ein Einfügen von Programmschritten einfacher und schneller möglich ist. Auch hat man die Möglichkeit, schon während der Programmierung Kommentare einzufügen, so daß man auch nach längerer Zeit seinen Entwurf noch verstehen und nachvollziehen kann. Nun aber zu den einzelnen Assemblern.

Mastercode

Mastercode ist ein kombinierter Editor/Assembler/Monitor und wird vom Verlag Markt & Technik AG vertrieben. Er ist auf Diskette und Kassette lieferbar.

Editor

Eine Äußerlichkeit hebt Mastercode von allen anderen Assemblern ab: Alle Funktionen werden über Menüs und Untermenüs abgewickelt. Dies ist gerade für einen Anfänger sinnvoll, der sich nicht eine Unmenge von Befehlen merken möchte.

Im Editor stehen die Unterpunkte Eingabe, Auflisten, Löschen, Umnumerieren, Speichern und Laden zur Verfügung. Dieser Editor ist völlig eigenständig, das heißt vom Basic-Editor unabhängig. Er arbeitet allerdings streng zeilenorientiert, so daß jede eingegebene Zeile mit einer Nummer beginnen muß. Man kann nicht, wie beim Editieren von Basic-Programmen, mit dem Cursor rauf und runter fahren.

Es lassen sich schon existente Bytefolgen in den Quelltext übernehmen, sie werden dann mit einem Pseudo-Opcode für Tabellen versehen. Dies ist aber nur nützlich bei frei verschieblichen Programmteilen, oder eben Tabellen, weil sonst alle Adressen von Hand angepaßt werden müßten.

Interessant ist noch, daß beim Laden von Quelltexten immer ein MERGE durchgeführt wird. Dadurch lassen sich recht einfach zwei Quelltextdateien mischen.

Der nutzbare Speicher für den Quelltext ist enttäuschend knapp gehalten. Der maximale Speicherbereich für Quelltexte beträgt nämlich 15 KByte, das entspricht maximal 1000 Zeilen Assemblerquelltext, Kommentare nicht eingerechnet. Selbst diese Zahl ist bei der Verwendung von vielen Labeln nicht erreichbar. Das wäre ja alles nicht so schlimm, wenn es möglich wäre, Quelltextfiles miteinander zu verketten. Ein nur etwas längerer Quelltext muß dann jedoch in zwei Einzeltexte aufgeteilt werden, wobei im zweiten sämtliche benötigten Label noch einmal definiert werden müssen, da beide Teile nur unabhängig voneinander assembliert werden können.

Auch die geringe Verarbeitungsgeschwindigkeit macht die Arbeit nicht gerade zum Vergnügen.

Assembler

Label dürfen bis zu sechs Zeichen lang sein und können auch, wenn man sie als Variable, das heißt als Zwischenspeicher benutzt, erst nach dem Gebrauch definiert werden. Es dürfen maximal 100 verschiedene Label im Quelltext auftreten.

Die Ausgabe der Symboltabelle, also die Auflistung sämtlicher verwendeter Label, kann nach Beendigung des Assemblierens erfolgen.

Mastercode unterstützt die Zahlensysteme binär, oktal, dezimal und hexadezimal sowie die Eingabe im ASCII-Code. Rechnungen dürfen im Quelltext in den vier Grundrechnungsarten durchgeführt werden. Tabellen lassen sich entweder ganz normal als Byte-für-Byte-Tabellen oder als Adreßtabellen aufbauen.

Es ist möglich, den Objectcode direkt in den Speicher oder auf ein Peripheriegerät (Kassette, Floppy) auszugeben oder den Quelltext einfach nur auf Fehler zu untersuchen. Auf

Im zweiten Teil unseres Assembler-Tests kommen die Assembler unter 100 Mark an die Reihe. Wir stellen folgende Produkte vor: Mastercode-Assembler, Profimat, Profisoft und Maschine 64. Obwohl sie in der unteren Preisklasse angesiedelt sind, können sie eine wertvolle Hilfe beim Einstieg in die Maschinensprache sein.

dem Drucker erhält man im Anschluß an die Assemblierung ein sauber formatiertes Quelltext/Objektcode-Listing.

Als letztes sei noch erwähnt, daß Kommentare nur für sich alleine in Quelltextzeilen stehen dürfen und nicht, wie sonst üblich, auch durch Semikolon getrennt hinter einem Assembler-Befehl.

Monitor

Mastercode hält nur die Minimalfunktionen eines Monitors bereit. Dazu gehören das Disassemblieren, der Hexdump und das Verändern von Speicherbereichen, das Starten von Maschinenprogrammen, das Laden und Speichern von Objektcode sowie ein einfacher Einzelschrittbetrieb zum Austesten von Maschinenprogrammen.

Bei einem Hexdump macht es anfangs richtig Spaß, zuzusehen, wie die einzelnen Bytes schön nacheinander auf dem Schirm erscheinen und schon nach einigen Sekunden ein paar Zeilen auf dem Bildschirm stehen. Aber schon bald beginnt die extrem geringe Geschwindigkeit aller Monitorfunktionen zu stören, da man nicht jedesmal minutenlang auf ein Disassemblerlisting oder sonstiges warten möchte. Ein kurzer Blick mit Mastercode auf Mastercode bestätigte dann den Verdacht, daß man es hier mit compiliertem Basic zu tun hat, das es wohl kaum mit irgendeinem anderen Monitor, der in Maschinensprache geschrieben wurde, aufnehmen kann.

Dokumentation

Als »Handbuch« erhält man knappe 50 Seiten im augenermüdenden Compactkassettenformat, was wohl von der ursprünglichen Vertriebsform auf Kassette herrührt. Inzwischen ist Mastercode aber auch auf Diskette erhältlich.

Dieses, aufgeklappt noch nicht einmal eine Postkarte bedeckende, Heftchen enthält aber, sauber gegliedert, alle Informationen, die zum Betrieb von Mastercode notwendig sind, diese allerdings etwas trocken und nicht gerade mit sinnvollen Beispielen aufgelockert. Im Anhang stehen Tabellen aller Opcodes des 6510 Mikroprozessors in alphabetischer wie numerischer Reihenfolge.

Als Fazit möchte ich ziehen, daß Mastercode sich wohl nur für diejenigen eignet, die kleine Aufgaben in Assembler lösen wollen und dabei nicht unter Zeitdruck stehen. Für einen Anfänger ist, wie schon erwähnt, die Menü-Struktur von Mastercode sehr nützlich.

Profimat

Das Assemblerpaket Profimat von Data Becker enthält den Assembler Profiass und den Monitor Profimon. Beschäftigen wir uns zunächst mit Profiass.

Assembler

Profiass-Quelltexte werden wie normale Basic-Programme eingegeben, können sogar in Basic-Programme eingebettet sein. Ein zusätzlicher Editor oder eine Befehlserweiterung zur leichteren Eingabe wird nicht mitgeliefert. Um gleich die Zusammenarbeit mit Basic-Programmen zu verdeutlichen: Direkt vor dem Quelltext muß Profiass mit SYS 32768 aufgerufen werden, ein Rücksprung ins Basic-Programm ist dann mit dem Pseudo-Opcode »GTB« möglich.

Label dürfen bei Profimat bis zu acht signifikante Zeichen haben. Berechnungen können in den drei üblichen Zahlensystemen (Hex., Dez., Bin.) vorgenommen werden. Hier sind die Grundrechenarten sowie logische Funktionen und Schiebeoperationen erlaubt. Man scheint allerdings das logische NOT vergessen zu haben, dafür ist XOR enthalten. Klammern dürfen beliebig gesetzt werden.

Eine interessante Eigenschaft von Profiass ist, erstellte Symboltabellen speichern und wieder laden zu können. Sollten Sie öfters Routinen aus dem Betriebssystem benötigen, brauchen Sie die entsprechenden Label nicht für jedes Programm neu zu definieren, sondern können sie in die Symboltabelle laden.

Ebenso interessant ist, daß mehrere Assemblerbefehle durch Doppelpunkt getrennt in einer Zeile stehen dürfen.

Zu den üblichen Pseudo-Opcodes für Tabellen tritt ».FLP« mit dem es möglich ist, Zahlen im Commodore-Fließkommaformat im Speicher abzulegen. Dies ist brauchbar, wenn man in Assembler Fließkommaroutinen schreiben möchte.

Quelltexte können miteinander verkettet werden, das heißt, ein Quelltext ruft den nächsten zu assemblierenden auf. Somit lassen sich auch größere Programme assemblieren, insbesondere, weil Profiass den Objektcode direkt zur Diskette schicken kann.

Auch der Ausdruck von Listings ist steuerbar und kann auf beliebige Peripheriegeräte gesandt werden. Diese Listings werden voll formatiert und sogar seitenweise mit Kopf- und Fußzeilen ausgedruckt.

Makros

Profiass bietet auch Makros. Ein Makro ist eine meist kurze Befehlssequenz, die man nicht jedesmal neu eintippen möchte und deshalb mit einem eigenen Namen versieht, mit dem sie dann jederzeit aufgerufen werden kann. Im späteren Objektcode steht dann anstelle des Makroaufrufes jedesmal die vorher definierte Befehlssequenz. Es ist auch möglich, dieser Befehlssequenz von Mal zu Mal wechselnde Parameter zu übergeben.

Makros bei Profimat dürfen beliebig viele interne Label verwenden. Diese müssen aber durch einen Punkt gekennzeichnet werden, dann erhalten sie automatisch eine Ordnungsnummer, so daß ein mehrmaliges Aufrufen des Makros möglich ist, ohne ein Label zweimal zu definieren. Zwei verschiedene Makros müssen aber verschiedene interne Label benutzen.

Makros dürfen hier nicht verschachtelt werden, das heißt ein Makro darf kein anderes Makro aufrufen.

Alle Makrodefinitionen müssen am Anfang des ersten zu assemblierenden Programms stehen, wenn Quelltexte verkettet werden sollen.

Damit wird das Aufbauen einer Makrobibliothek sehr erschwert.

Insgesamt sind gegenüber anderen, teureren Assemblern die Möglichkeiten der Makros eingeschränkt und wenig flexibel. Meiner Meinung nach hätte dieser Programmteil ruhig einer besseren Benutzerführung zum Opfer fallen können.

Monitor

Der Monitor Profimon enthält so ziemlich alle üblichen Befehle. Es können die Register sowie Speicherstellen in Hex-Dumps oder disassembliert angezeigt werden. Speicherbereiche können verschoben, mit anderen Bereichen verglichen oder mit bestimmten Werten gefüllt werden. Auch das Durchsuchen nach einer Bytefolge ist möglich, ebenso wie natürlich Laden und Speichern von Programmen. Zum Austesten von Programmen ist ein Breakpoint frei definierbar, bei dessen Erreichen in den vorhandenen Einzelschrittmodus umgeschaltet wird.

Mich stört allerdings, daß hier kein einfacher Assembler wie in anderen Monitoren vorhanden ist. Es ist eigentlich zu umständlich, für jede kleine Änderung im Programm den Quelltext erneut zu assemblieren, insbesondere, da Profimon ja eigentlich ein eigenständiges Programm sein soll. Eine Eigenschaft ist allerdings ungewöhnlich. Mit Profimon kann man auch auf das RAM unter dem Basic- und Kernal-ROM zugreifen oder das Charakter-ROM auslesen. Insgesamt gesehen, läßt es sich mit Profimon nicht anders als mit anderen handelsüblichen Monitoren arbeiten.

Dokumentation

Es befinden sich 34 Seiten Anleitung in einem DIN-A5-Ordner im typischen Data Becker-Design. Diese Anleitung hat aber einige Mängel. So wird man beispielsweise erst bei der Erklärung der Verkettung von Quelltexten mit der Tatsache überrumpelt, daß der Assembler mit SYS 32768 gestartet wird, was vorher nicht erwähnt wurde. Wer gezielt Informationen sucht, wird sie teilweise nicht finden, da sie manchmal in thematisch anderen Abschnitten versteckt sind. Das ist eigentlich schade, weil gerade auch der Profimat wohl die Erstkäufer eines Assemblers anspricht. Der Profimat ist nach Preis und Qualität für den Anfänger mit Aufstiegsambitionen geeignet. Für fortgeschrittene Programmierer bietet er allerdings zu wenig.

Profisoft-Assembler

Profisoft vertreibt ein kleines Programmpaket zur Erstellung von Assemblerprogrammen, das im folgenden in Ermangelung eines klangvollen Namens mit Profisoft-Assembler bezeichnet wird. Dieses Programmpaket umfaßt neben einem Assembler auch einen Re-Assembler, einen Mini-Monitor sowie einen Editor und ist sowohl auf Kassette als auch Diskette erhältlich.

Editor

Als Editor für den Quelltext wird der normale Basic-Editor verwendet. Um die Eingabe komfortabler zu gestalten, wurden einige Toolkit-ähnliche Funktionen in das Programm eingebaut. Diese stehen dann auch zum Schreiben und Editieren von Basic-Programmen zur Verfügung.

Insgesamt gibt es 16 neue Befehle, die alle mit einem »←« beginnen. Darunter befinden sich neben den Aufrufbefehlen für Assembler, Re-Assembler und Monitor, Befehle für automatische Zeilennumerierung und Zeilenumnumerierung (Renumber). Es ist ein Re-New vorhanden, das auch nach einem Reset wirksam ist und den Quelltext, sofern er nicht anderweitig zerstört wurde, wieder regeneriert. Dies ist besonders in der Testphase von Programmen nützlich, da man sich das ewige Abspeichern und Neuladen erspart.

Ein weiterer interessanter Befehl ist »←F«, mit dem man nicht nur Zeichenketten im Quelltext finden, sondern auch durch andere ersetzen kann. Ebenso ist das SAVEn bestimmter Speicherbereiche möglich, wie auch die Ausgabe der genauen Speicherbelegung der Quelltexte und Tabellen. Alles in allem ist das Erstellen von Quelltext sehr komfortabel und dürfte wohl auch höheren Ansprüchen genügen.

Assembler

Doch nun zum Assembler selbst. Hier darf ein Label aus maximal acht Zeichen bestehen, dabei sollten allerdings keine Leer- und Sonderzeichen verwendet werden. Tabellen werden mit drei Pseudo-Opcodes unterstützt, einer für Einzelbytes, einer für Texte und einer für Adressen. Während des zweiten Durchlaufs kann ein Listing wahlweise auf Bildschirm oder Drucker (nach OPEN) ausgegeben werden. Dieses Listing wird allerdings nicht formatiert, das heißt Labels und Kommentare stehen nicht geordnet untereinander. Dies läßt sich nur durch entsprechende Eingabe des Quelltextes erreichen. Einzelne Quelltextfiles auf Diskette/Kassette können beliebig aneinandergehängt beziehungsweise nachgeladen werden.

Re-Assembler

Ein sehr nützlicher Programmteil ist der eingebaute Re-Assembler. Dieser wird in der Anleitung fälschlicherweise als Disassembler bezeichnet. Im Gegensatz zu einem Disassembler, der Maschinenprogramme auf dem Bildschirm anzeigt oder auf dem Drucker ausgibt, erzeugt ein Re-Assembler wieder Quelltext, der dann nach Änderungen erneut assembliert werden kann. Der Vorteil liegt auf der Hand. Wenn Sie nachträglich in ein Programm etwas einfügen oder es umschreiben wollen, aber den Quelltext nicht besitzen, so können Sie sich diesen einfach regenerieren. Natürlich ist dieser Quelltext nicht gleich dem Original, denn woher sollte der Re-Assembler beispielsweise die Namen der einzelnen Labels kennen?

Bei einem vom Profisoft-Re-Assembler erzeugten Quelltext werden alle Speicherzugriffe und Sprünge über Labels abgewickelt. Um die Labels eindeutig zu halten, sehen sie folgendermaßen aus: Der erste Buchstabe ist ein L, gefolgt von der hexadezimalen Adresse des tatsächlichen Speicherplatzes, zum Beispiel LFFD2. Sollte der Re-Assembler auf einen undefinierten Opcode stoßen, so behandelt er ihn als Mini-Tabelle, bestehend aus einem Byte mit dem entsprechenden Pseudo-Opcode. In einem Durchlauf können maximal 4 bis 6 KByte re-assembliert werden; er dauert zirka 30 Sekunden.

Der Mini-Monitor

Nun noch einige Worte zum Mini-Monitor. Mit ihm lassen sich Speicherbereiche hexadezimal anzeigen, verändern und verschieben. Damit sind seine Möglichkeiten schon ausgeschöpft. Er ist allerdings insbesondere für das Verschieben des erzeugten Object-Codes notwendig, wenn der Quelltext an eine andere Stelle als der späteren tatsächlichen Startadresse assembliert wurde.

Dokumentation

Ein Programm, das gerade wegen seines Preises und seiner angepaßten Leistungen für Anfänger und

leicht Fortgeschrittene geeignet ist, benötigt natürlich auch eine gute Anleitung. Die vorliegende ist mit 16 Seiten DIN A5 etwas knapp. Sie setzt schon Kenntnisse der Maschinensprache und des Computers voraus. Es werden alle Funktionen hinreichend genau erklärt. Leider sind nur sehr wenige Beispiele abgedruckt. Ist man allerdings erst einmal mit dem Programm vertraut, wird man die Anleitung gerne weiterhin als schnelles Nachschlagewerk benutzen.

Insgesamt hat das Programm einen sehr guten Eindruck hinterlassen. Seine Bedienung ist einfach und doch komfortabel. Auch bei Fehlern stürzt der Profisoft-Assembler nicht ab (ein Totalabsturz ist nur bei mutwilligem Überschreiben des Assemblers selbst möglich). Mit dem Re-Assembler können auch fremde Programme leicht verändert werden. Das fordert direkt zum Experimentieren heraus. Ein kleiner Schwachpunkt ist lediglich der Monitor.

Maschine 64

Der letzte hier vorgestellte Assembler heißt Maschine 64 und wird von Dynamics vertrieben. Auch Maschine 64 erfüllt eine Vielzahl von Funktionen, es ist Assembler, Re-Assembler, Toolkit, DOS-Support, Monitor und Disk-Monitor in einem, und das bei nur 16 KByte Speicherverbrauch.

Editor

Genauso wie beim Profisoft-Assembler werden Quelltexte wie Basic-Programme eingegeben. Die Eingabe des Quelltextes wird hier von einem Toolkit unterstützt. Vorhanden sind hier die Befehle AUTO, APPEND, DELETE, FIND, RENUMBER. Zusätzlich gibt es dann noch die Befehle ASSEMBLER, REASSEMBLE, MONITOR, DISKMONITOR und BYE, welche die anderen Teile des Systems aufrufen, beziehungsweise Maschine 64 abschalten.

Alle diese Befehle sind auch zum Editieren ganz normaler Basic-Programme geeignet. Zusätzlich ist im Editor, wie auch in allen anderen Programmteilen, eine Diskettenunterstützung, ähnlich dem DOS 5.1, eingebaut.

Assembler

Über den Assembler selbst läßt sich folgendes sagen: Label dürfen beliebig lang sein, es werden allerdings nur die ersten 20 Zeichen unterschieden.

Sehr komfortabel ist das Anlegen von Texttabellen. Diese können nicht nur im ASCII-Code, wie üblich, sondern auch im Bildschirmcode und im invertierten Bildschirmcode angegeben werden. Dies ist sehr nützlich, wenn man Texte direkt in den Bildschirmspeicher schreiben und nicht über die Betriebssystemroutine »Zeichen ausgeben« arbeiten will.

Maschine 64 erlaubt auch Berechnungen im Quelltext. Diese sind allerdings auf Addition und Subtraktion sowie LO-Byte und HI-Byte-Isolierungen beschränkt. Es dürfen maximal acht Klammern gesetzt werden. Das Assemblerlisting am Ende des zweiten Pass wird teilweise formatiert und kann, ebenso wie die Symboltabelle, auch auf einem Drucker ausgegeben werden.

Re-Assembler

Der Re-Assembler läßt kaum Wünsche offen. Er erzeugt aus Objektcode wieder Quelltext. Auch hier werden, soweit wie möglich, Sprünge und Speicherzugriffe über Label abgewickelt. Besonders komfortabel sind allerdings drei Punkte: Dem Re-Assembler kann vor dem Start mitgeteilt werden, wo Tabellen und wo tatsächliches Programm im Speicher stehen, so daß Tabellen und Programm in einem Arbeitsgang in Quelltext umgewandelt werden können. Sollte der Re-Assembler auf einen nicht als Programm identifizierbaren Bytewert treffen, wird aus ihm eine Ein-Byte-Mini-Tabelle mit angehängtem ERROR. Solche Zeilen können dann sehr schnell mit dem FIND-Befehl ausfindig gemacht werden. Als letztes ist es auch möglich, einen bestimmten Speicherbereich so zu reassemblieren, als ob er in einem anderen Bereich stehen würde.

Monitor

Besitzern des PET, dem Großvater der Home- und Personal Computer, wird der eingebaute Monitor unter dem Namen SUMO bekannt sein. Er kann Speicherbereiche anzeigen und ändern sowie durchsuchen, verschieben, vergleichen, füllen, laden und speichern. Ein Disassembler ist ebenso vorhanden wie auch ein einfacher Line-by-Line-Assembler, der sich gerade bei kleinen Änderungen an einem Programm bezahlt macht. Auch sind Umrechnungen Dez-Hex und umgekehrt möglich. Leider fehlt hier ein Einzelschrittmodus, mit dem man Maschinenprogramme Schritt für Schritt auf Funktionstüchtigkeit testen kann.

Disk-Monitor

Kurz angesprochen werden soll auch noch der Disk-Monitor. Beliebige Blöcke der Diskette können in den C 64-Speicher geholt und angezeigt, verändert und wieder zurückgeschrieben werden. Dabei ist es dann auch möglich, innerhalb eines Blocks Bytefolgen zu suchen oder zu verschieben oder gar den Block zu disassemblieren. Auch hier funktioniert dann der Line-by-Line-Assembler. Leider fehlen Befehle, mit denen man blockübergreifend arbeiten könnte, so daß man immer auf einen einzelnen Block bei der Arbeit fixiert ist.

Dokumentation

Die knapp 30 Seiten Anleitung im DIN-A5-Format machen im ersten Augenblick einen recht guten Eindruck, doch werden hier einige nützliche Details der einzelnen Programme verschwiegen, die man erst beim Probieren durch Zufall herausfindet. Ansonsten ist der Text sehr locker und leicht zu lesen. Leider sind aber auch hier die Beispiele sehr knapp gehalten. Ein Fehler im Handbuch soll nicht unerwähnt bleiben. Entgegen der Aussage, daß sich das Programm selbst starte, mußte RUN eingegeben werden. Das kann einen beim ersten Kontakt mit Maschine 64 doch leicht in Verwirrung bringen.

Insgesamt gesehen ist auch Maschine 64 ein sehr brauchbares, sicheres und bedienerfreundliches Programm. Gerade seine vielen kleinen Details und Zusatzfunktionen erleichtern die Arbeit ganz erheblich. Mit Maschine 64 wird man auch als fortgeschrittener Programmierer nicht so schnell den Wunsch nach einem anderen Assembler verspüren. (B. Schneider/gk)

Fazit

Die vier hier vorgestellten Assembler bieten alle eine dem Preis entsprechende Leistung. Man sollte sich jedoch vor einem Kauf genau überlegen, wie weit man in die Maschinensprache einsteigen will. Zum »Reinschnuppern« in die Assemblerprogrammierung genügen diese Programme allemal.

Mastercode Assembler: Markt & Technik, Hans-Pinsel-Str. 2, 8013 Haar bei Munchen, Preis: Kassette 48 Mark, Diskette 63 Mark

Profimat: Data Becker, Merowinger Str. 30, 4000 Düsseldorf, Preis: Diskette 99 Mark

Profisoft-Assembler: Profisoft, Sutthauser Str. 50-52, 4500 Osnabrück, Preis: Kass/Disk. 75 Mark

Maschine 64: Dynamics, Postfach 112005, 2000 Hamburg 11, Preis: Diskette 79 Mark

Basic-Programm auf TRAB

Haben Sie schon einmal daran gedacht, sich eventuell einen Compiler zuzulegen, um Basic-Programme schneller zu machen? Sie wissen aber vielleicht noch zuwenig über Compiler, um sich den richtigen auszusuchen. Wir stellen Ihnen deshalb vier Typen vor.

Wir haben vier der bekanntesten Compiler getestet: Petspeed, Austro-Speed, BASS- und Ex-Basic Level II-Compiler. Bevor wir uns jedoch mit den »Prüflingen« genauer beschäftigen, wollen wir erst ein wenig auf die praktischen Grundlagen der Compiler eingehen.

Wenn man Compiler hört, denkt man unwillkürlich immer zuerst an den Geschwindigkeitsgewinn, den diese Programme bringen sollen. In der Tat ist der Hauptzweck von Compilern in der zeitlichen Optimierung eines Programmlaufs zu sehen, damit die Ausführungszeit verkürzt wird. So gibt es demnach viele Compiler, deren Hauptkonzept in der Geschwindigkeitserhöhung liegt.

Über diese Tatsachen darf man aber auch andere Eigenschaften von Compilern nicht vergessen. Compiler erhöhen zwar in der Regel die Geschwindigkeit der Basic-Programme, sie vertragen aber unter Umständen gewisse syntaktische Konstruktionen nicht, die beim Interpreter ohne weiteres funktionieren. Außerdem werden kleinere Programme in der Regel durch das Compilieren um einiges länger als sie ursprünglich waren.

Da wir gerade bei der Programmlänge sind, soll an dieser Stelle auch gleich auf die zwei verschiedenen Arten von Compilern eingegangen werden. Die erste Art ist der Assemblercode-Compiler, der echten Maschinencode erzeugt und damit maximalen Geschwindigkeitsgewinn bringt. Der Nachteil dieser Methode ist, daß das compilierte Programm (Compilat) in der Regel

Hersteller	Austro-Speed	BASS	Exbasic Level II-Compiler	Petspeed
Preis (ca.)	298,—	198,—	298,—	149,—
Lieferumfang	1 Diskette 1 Handbuch	3 Disketten 2 dicke Handbücher	1 Diskette 1 Handbuch	1 Diskette 3 Seiten Einweisung
Dokumentation	gut	ausgezeichnet	befriedigend	mangelhaft
ungefähre Länge des Programms	63 Blocks	263 Blocks	290 Blocks	300 Blocks
Anzahl PASSES	2	2 + 2 Assembler (wird nicht mitgeliefert)	2 + 2 Assembler (integriert)	4
Programmschutz möglich?	ja	nein	nein	nein
Integer Arithmetik	ja	ja	ja	ja
Erweiterungen möglich?	ja	ja	ja	nein
variables DIM möglich?	ja	nein	nein	nein
Automatisches DIM auf 11 Elemente	ja	nein	nein	ja
Compilierdauer EDDI (14 Blocks)	3 min	7,10 min + 9 min ASSI	12 min	7,30 min
Diskettenwechsel beim Compilieren	nein	ja, 2mal	ja, 4mal	nein
Anzahl der erzeugten Files	2	10	1	2
Länge des Compilats (EDDI)	32 Blöcke	39 Blöcke	39 Blöcke	42 Blöcke
Compilertyp	—	Adreßcode + Assemblercode	Assemblercode	Assemblercode

Tabelle 1. Vier Compiler im Vergleich

			Basic	Ex- + BASS	Petspeed	Austro-Speed
Test	1a	Einlesen (100 Werte)	1,53	0,67	0,50	0,46
Test	1b	Sortieren	64,16	24,85	8,23	16,25
Test	1c	Anzeigen	0,55	0,67	0,10	0,33
Test	1	Gesamt (a + b + c)	66,25	26,18	8,83	17,05
Benchmark	1	FOR NEXT	1,83	1,01	0,28	1,10
	2	430 K = K + 1 440 IF K < 1000 THEN 430	13,22	2,16	0,72	1,38
	3	A = K/K x K + K − K	11,25	4,70	5,54	4,38
	4	A = K/2 x 3 + 4 − 5	12,00	6,51	6,42	5,32
	5	GOSUB RETURN	16,65	0,45	0,18	0,15
	6	FOR L = 1 TO 5:NEXT L	14,47	6,34	1,93	6,50
	7	FOR L = 1 TO 5:M(L) = A:NEXT	24,85	8,80	2,50	3,95
	8	A = K↑2 B = LOG(K) C = SIN(K)	112,30	102,19	93,25	101,98
Gesamtzeit (Test 1 + Bench 1 bis 8)			390,86 ≙100%	198,53 ≙ 50,79%	145,52 ≙ 37,23%	172,96 ≙ 44,25%

Tabelle 2. Die vier Compiler im Zeitvergleich. Im Test 1 wurden mit der RND-Funktion 100 Zeichen ermittelt, sortiert und nebeneinander ausgegeben. Die Benchmark-Tests 1 bis 8 waren so aufgebaut, daß die Zeit für die angegebenen Befehle selbst ermittelt werden konnten. Deshalb ist die Gesamtzeit nicht identisch mit der Summe der einzelnen Testzeiten. Jeder Befehl (Benchmark 1 bis 8) wurde 1000mal durchgeführt (siehe Listing 1).

gebracht – Compiler im Test

um einiges länger wird als das ursprüngliche Basic-Programm.

Bei der zweiten Art von Compilern (sogenannte Adreßcode-Compilern) geht man deshalb einen anderen Weg. Hier wird der Basic-Text in eine Liste von Sprungadressen umgewandelt, was Speicherplatz spart, aber andererseits wieder auf Kosten der Geschwindigkeit geht.

Wie Sie sehen, gibt es den perfekten Compiler nicht. Entweder ist ein Programm sehr schnell, dann ist es länger, oder ein Programm ist nicht ganz so schnell, dafür wird es kürzer. Einen guten Compiler erkennt man also am richtigen Kompromiß zwischen Geschwindigkeit und Länge des Compilats. Ein weiteres Qualitätsmerkmal für Compiler ist die Dauer des Compilierens. Dieser Vorgang ist zwar in der Regel einmalig, da man nur ein wirklich fertiges Programm compilieren wird, er sollte sich dennoch in vernünftigen Grenzen bewegen.

Wenn wir uns jetzt gleich einmal mit den speziellen Eigenschaften der Testkandidaten vertraut machen, sollten Sie immer an diese Merkmale denken. Das interessante an den getesteten Compilern ist nämlich, daß sie fast alle nach unterschiedlichen Kriterien entwickelt wurden.

Der Austro-Speed-Compiler

Der erste Compiler, mit dem wir uns beschäftigen wollen, ist der Austro-Speed von Commodore. Austro-Speed ist eine verbesserte Version des Austro-Comp für die CBM-Systeme. Im Lieferumfang sind eine Diskette mit dem Programm, ein Handbuch mittleren Umfangs und ein Dongle enthalten. Ein Dongle ist ein programmschutztechnischer Hardwarezusatz, dessen Vorhandensein abgefragt wird und ohne das der Compiler nicht läuft, (in diesem Fall ein Stecker für den User-Port). Als ich mir das Inhaltsverzeichnis der Diskette ansehen wollte, erlebte ich sofort eine Überraschung. Der ganze Compiler besteht aus einem einzigen Programm mit einer Länge von 63 Blöcken (15,75 KByte).

Als Testprogramm für den Compiliervorgang diente der Disk-Monitor EDDI aus der 64'er, Ausgabe 10/1984. Dieses Programm hat den Vorteil, daß es nicht zu kurz ist. Außerdem ist der Programmierstil an vielen Stellen alles andere als sauber. Haben Sie übrigens bemerkt, daß sich bei EDDI ein Fehler eingeschlichen hat? In der Zeile 1070 ist die IF-Abfrage überflüssig und zeigt außerdem auf eine nicht vorhandene Zeile (1 090). Diese Abfrage stört den Programmablauf nicht im geringsten, wir können jedoch gespannt auf die Reaktionen der Compiler sein, wenn sie diese Zeile abarbeiten. Jetzt aber wieder zurück zu Austro-Speed.

Bevor man den Compiler in den Computer lädt, muß man darauf achten, daß das Dongle auf den User-Port des C 64 gesteckt ist. Für den Vorgang des Compilierens ist es in der Regel notwendig, daß viel Platz auf der Diskette mit dem Basic-Programm vorhanden ist, da die Compiler eine Menge Dateien erstellen, die jedoch nach dem Compilieren normalerweise wieder gelöscht werden. Das ist besonders bei langen Programmen zu beachten.

Nachdem wir auch diese letzte Vorbereitung ausgeführt haben, starten wir den Compiler. EDDI besteht aus 14 Blöcken auf Diskette. Nach genau drei Minuten ist Austro-Speed mit der Arbeit fertig, und das Compilat steht zur Verfügung.

Während des Compilierens hat Austro-Speed sogar den Programmfehler entdeckt und angezeigt, jedoch seine Arbeit nicht unterbrochen.

Wenn wir uns die Diskette betrachten, so sehen wir, daß unser Programm an Länge ganz erheblich zugenommen hat. Es besteht jetzt aus 32 Blöcken und ist damit mehr als doppelt so lang geworden.

Zur Erleichterung einer eventuell noch folgenden Korrektur bei einem Fehler, legt Austro-Speed noch ein weiteres File ab, das die neuen Speicheradressen sämtlicher Programmzeilen enthält.

Da ein Compiler ein Basic-Programm nicht auf einmal übersetzt, sondern dafür mehrere Durchläufe benötigt, kann man auch anhand der Anzahl dieser Durchläufe (Durchlauf = Pass) einen Compiler charakterisieren. Austro-Speed benötigt für seine Arbeit zwei dieser Durchläufe; man bezeichnet ihn deshalb auch als 2-Pass-Compiler.

Unser nächstes Programm heißt BASS und kommt von gmbsoft. Der Unterschied zu Austro-Speed wird sofort deutlich, wenn man sich den Lieferumfang betrachtet. Er besteht aus drei Disketten und zwei dicken Handbüchern. Bei einer der Disketten handelt es sich um eine Demodiskette, die unter anderem ein Sortierprogramm enthält, um die Geschwindigkeit eines Compilats zu demonstrieren.

Der BASS-Compiler

Wie bei Austro-Speed habe ich auch hier erst einmal das Directory gelistet. Hat man die Länge der Austro-Speed noch vor Augen, so trifft einen hier fast der Schlag. Bei BASS handelt es sich um den reinsten »Mammutcompiler«. Er arbeitet zwar auch nur mit zwei Durchläufen, jedoch besteht hier allein schon der Pass 1 aus einem über 100 Blöcken langen Programm, der von Pass 2 noch übertroffen wird.

Da es bei einer solchen Komplexität kaum möglich ist, einfach »drauflos« zu arbeiten, sollte man sich erst einmal eines der beiden Handbücher vornehmen (das dünnere, versteht sich). Und hier erlebt man auch gleich die erste Überraschung. Der Compiler erzeugt bei seiner Arbeit kein lauffähiges Programm, sondern nur eine stattliche Anzahl von Dateien (insgesamt 10), die editierfähig sind und noch einen Assemblierlauf benötigen, bevor ein fertiges Programm daraus entsteht.

Diese Konzipierung hat aber natürlich einen Sinn. Bei gmbsoft hat man versucht, einen Compiler herzustellen, der so offen wie möglich arbeitet; das heißt der Programmierer soll auch nach dem Compilieren noch die Möglichkeit haben, optimierende Eingriffe und Änderungen an seinem Programm vorzunehmen. Zu diesem Zweck eignet sich ein editierfähiger Code natürlich eher, als der »Spaghetticode« in einem fertig compilierten Programm.

Durch diese sehr positive Eigenschaft des Programms angeregt, geht man erneut an die Arbeit, aber — wo ist der Assembler?

Es stellt sich heraus, daß der Assembler nicht im Lieferumfang des BASS enthalten ist; er muß extra besorgt werden. Wie im Handbuch empfohlen, beschafft man sich also das Assemblerpaket ASSI von Dirk Zabel (siehe Assembler-Test in Aus-

gabe 1/85), um endlich ein fertiges Compilat zu erhalten.

Den gewohnten Richtlinien folgend kopiert man das Testprogramm EDDI auf eine leere Diskette. Aber es müssen noch einige Handgriffe ausgeführt werden, bis das Programm endlich fertig compiliert sein wird. Zuerst muß noch eine Bibliothek auf die Programmdiskette kopiert werden, die der Assembler benötigt, und dann kann es endlich losgehen. Da BASS, wie schon erwähnt, ziemliche Ausmaße besitzt, lag es natürlich nahe, einmal Hypra-Load heranzuziehen, und siehe da — BASS arbeitet mit Hypra-Load einwandfrei zusammen, was die Compilierzeit insgesamt erheblich verkürzt.

Trotz aller dieser »Vorabhandgriffe« entpuppt sich der BASS als Langweiler. Für das reine Compilieren von EDDI benötigte er 7,10 Minuten. Das nachfolgende Assemblieren benötigt noch einmal neun Minuten, so daß man insgesamt mindestens eine halbe Stunde beschäftigt ist (alles eingerechnet).

EDDI wird von BASS einwandfrei verarbeitet; der Fehler in Zeile 1070 wurde jedoch während der Compilation nicht entdeckt. Er wurde erst vom Assembler registriert und äußerte sich in einem »UNDEFINED SYMBOL ERROR«. Auch in diesem Fall wurde die Arbeit jedoch ordnungsgemäß zu Ende geführt.

Der Exbasic Level II-Compiler

Der Exbasic Level II-Compiler von Interface Age machte bei Erhalt der Lieferung wieder einen ganz anderen Eindruck als der BASS. Diese beiden Compiler sind dabei fast identisch. Was den Namen dieses Compilers betrifft, so erscheint er vielleicht etwas irreführend. Der Exbasic Level II-Compiler hat mit Exbasic Level II ebensoviel oder ebensowenig zu tun, wie fast alle anderen Compiler dieses Tests auch.

Der Name soll eine Eigenschaft dieses Compilers verdeutlichen, die Austro-Speed und BASS jedoch ebenso besitzen: die Verarbeitung von Erweiterungen (sogenannte Extensions).

Das heißt nichts weiter, als daß diese Programme in der Lage sind, auch Befehle, die im Standard-Basic V 2.0 von Commodore nicht vorkommen, zu verarbeiten. Wenn diese Compiler zum Beispiel auf einen Befehl des Exbasic Level II stoßen, so wird dieser Befehl nicht compiliert, sondern im non-compiled-Code angelegt. Wird ein so compiliertes Programm jetzt zum Beispiel unter Exbasic gestartet, so übergibt das Steuerprogramm, das jedes Compilat enthält, den entsprechenden Befehl einfach dem Interpreter zur Ausführung und macht anschließend weiter.

Der Unterschied zwischen dem Exbasic Level II-Compiler und BASS besteht lediglich in der Dicke des Handbuchs, im Preis und in der Tatsache, daß der Exbasic-Compiler um den notwendigen Assembler erweitert wurde. Auf der Diskette erkennt man das an Pass 3 und Pass 4, die der BASS nicht besitzt.

Im Test zeigte der Exbasic-Compiler demzufolge auch die gleichen Eigenschaften wie der BASS, auf die ich gleich noch zu sprechen komme.

Das Compilieren und Assemblieren wird vom Exbasic Level II-Compiler um einiges schneller erledigt, als von BASS. Außerdem spart man sich das Kopieren der Bibliothek. Für EDDI wurde eine Zeit von 12 Minuten gemessen, was jedoch immer noch viermal so lang ist, wie beim Austro-Speed. Durch das jeweilige Nachladen der einzelnen Programmteile ergibt sich außerdem noch zusätzlich ein viermaliger Diskettenwechsel.

Da der BASS- und Exbasic Level II-Compiler nahezu identisch sind, soll auch gleich einmal auf die negativen Seiten der beiden Programme eingegangen werden.

Wie Sie vielleicht wissen, kann man im Commodore-Basic sowohl mit Fließkomma- als auch mit Integerwerten rechnen. Der Unterschied zeigt sich in den Variablennamen, wobei die Integervariablen durch ein »%« am Ende gekennzeichnet sind. Die Integerarithmetik läßt nur Zahlenbereiche von —32768 bis 32767 zu; hat aber dadurch den Vorteil, daß weniger Speicherplatz und geringerer Zeitaufwand beim Rechnen mit diesen Werten erforderlich ist. Im Gegensatz zu Fließkommawerten benötigen Integerzahlen normalerweise nur 2 Byte Speicherplatz pro Wert; das sind 3 Byte weniger als bei Fließkommaberechnungen.

Das Commodore-Basic hat jetzt den Nachteil, daß keine Integerroutinen existieren, die die Berechnungen durchführen. Alle Zahlen werden deshalb zuerst ins Fließkommaformat umgewandelt und dann verrechnet. Anschließend konvertiert der Interpreter diese Werte wieder zurück.

Alle Vorteile, die die Integerzahlen also haben, werden durch den Interpreter zunichte gemacht. Die Entwickler von Compilern haben dieses Manko sehr wohl erkannt, und deshalb sind alle getesteten Produkte auch mit eigenen Integerroutinen ausgestattet, die einen enormen Zeitgewinn versprechen. Es ist somit möglich, auch beim Interpreter verbotene Konstruktionen, wie eine Integerschleife, zu verwenden. FOR X% = 0 TO 1000 : NEXT X% ist zum Beispiel beim Interpreter nicht gestattet und wird mit einem »SYNTAX ERROR« quittiert.

Oben wurde schon auf Nachteile vom BASS- und Exbasic Level II-Compiler hingewiesen. Bei diesen beiden Produkten gibt es eine solche Konstruktion ebenfalls nicht. Hier muß man alle Variablen, die man als Integer verwenden möchte, mit direkten Befehlen an den Compiler als solche vordefinieren.

Als weiterer Minuspunkt zeigte sich bei diesen beiden Produkten die »Intoleranz« gegenüber der Syntax von Programmen.

Bei Basic-Programmen ist es üblich, eine Dimensionierung von Variablen, sofern das Feld nicht mehr als elf Elemente benötigt, zu unterlassen. Der Interpreter übernimmt diese Dimensionierung automatisch. Bei besagten beiden Compilern ist dies jedoch nicht der Fall und führt während des Compilierens zu einer Fehlermeldung in Form einer Nummer. Da diese beiden Compiler jedoch mit einer Fülle an Fehlermeldungen ausgestattet sind, erwies es sich bei dem Exbasic-Compiler als äußerst nachteilig, daß er keinen Fehlertext, sondern nur die Nummer der Meldung ausgibt. Wie es sich zeigte, enthält das Handbuch zwar eine Aufstellung aller Fehlermeldungen; diese aber wiederum ohne Nummer (im Gegensatz zum BASS), so daß man spekulativ schon sehr auf Zack sein muß, um zu erfahren, was für ein Fehler denn nun beanstandet wurde.

Auch das oft übliche Belegen einer Zeile mit einem Doppelpunkt »:«, um ein Programm lesbarer zu gestalten, wurde nicht toleriert und führte zu einer Fehlermeldung.

Insgesamt also eine Reihe von Nachteilen, die einem die Arbeit mit einem Compiler sicherlich schwerer machen, zumal wir an den anderen Testkandidaten feststellen konnten, daß es auch anders geht. Austro-Speed ist syntaktisch sehr großzügig. Das einzige, was er und Petspeed nicht vertragen, sind ver-

```
10 INPUT"WIEVIEL WERTE";Q1                      <044>
20 DIM FF$(1000)                                <092>
30 T1=TI                                        <242>
40 FOR I=1 TO Q1                                <251>
50 FF$(I)=CHR$(INT(RND(0)*26+64))               <176>
60 NEXT                                         <190>
70 T2=TI                                        <027>
80 GOSUB 930                                    <121>
90 T3=TI                                        <048>
100 PRINT:PRINT                                 <208>
110 FOR I=1 TO Q1                               <065>
120 PRINT FF$(I);                               <150>
130 NEXT                                        <004>
140 T4=TI                                       <099>
150 GOSUB 330                                   <185>
160 PRINT:PRINT                                 <012>
170 PRINT"ANZAHL WERTE= "Q1:PRINT               <094>
180 PRINT"EINLESEZEIT={3SPACE}"(T2-T1)/60       <041>
190 PRINT"SORTIERZEIT={3SPACE}"(T3-T2)/60       <088>
200 PRINT"ANZEIGEZEIT={3SPACE}"(T4-T3)/60       <063>
210 PRINT"GESAMT ZEIT={3SPACE}"(T4-T1)/60       <005>
215 PRINT:PRINT                                 <067>
220 PRINT"BENCH1{5SPACE}={3SPACE}"(B1-B0)/60
                                    <123>
230 PRINT"BENCH2{5SPACE}={3SPACE}"(B2-B1)/60
                                    <136>
240 PRINT"BENCH3{5SPACE}={3SPACE}"(B3-B2)/60
                                    <149>
250 PRINT"BENCH4{5SPACE}={3SPACE}"(B4-B3)/60
                                    <162>
260 PRINT"BENCH5{5SPACE}={3SPACE}"(B5-B4)/60
                                    <176>
270 PRINT"BENCH6{5SPACE}={3SPACE}"(B6-B5)/60
                                    <189>
280 PRINT"BENCH7{5SPACE}={3SPACE}"(B7-B6)/60
                                    <202>
290 PRINT"BENCH8{5SPACE}={3SPACE}"(B8-B7)/60
                                    <215>
300 PRINT"GESAMT ZEIT={3SPACE}"(B8-T1)/60      <082>
310 IF F=1 THEN PRINT:PRINT:PRINT:PRINT:PRINT#1
 :CLOSE 1:END                                  <108>
320 OPEN 1,4:CMD 1:F=1:GOTO 170                <055>
330 REM-------------------------------         <198>
340 REM     BENCHMARKS                         <194>
350 REM--------------------1-------            <222>
360 REM                                        <248>
370 B0=TI                                      <052>
380 FOR K=1 TO 1000                            <145>
390 NEXT K                                     <084>
400 B1=TI                                      <083>
410 REM--------------------2---------          <117>
420 K=0                                        <210>
430 K=K+1                                      <210>
440 IF K<1000 THEN 430                         <065>
450 B2=TI                                      <134>
451 REM--------------------3---------          <159>
452 K=0                                        <242>
453 K=K+1                                      <233>
454 A=K/K*K+K-K                                <223>
455 IF K<1000 THEN 453                         <085>
456 B3=TI                                      <141>
460 REM--------------------4---------          <169>
470 K=0                                        <004>
480 K=K+1                                      <004>
490 A=K/2*3+4-5                                <165>
500 IF K<1000 THEN 480                         <130>
510 B4=TI                                      <196>
520 REM--------------------5---------          <231>
530 K=0                                        <065>
540 K=K+1                                      <065>
550 A=K/2*3+4-5                                <226>
560 GOSUB 600                                  <085>
570 IF K<1000 THEN 540                         <198>
580 B5=TI                                      <012>
590 GOTO 620                                   <113>
600 RETURN                                     <232>
610 REM--------------------6---------          <066>
620 K=0                                        <155>
630 K=K+1                                      <155>
640 A=K/2*3+4-5                                <060>
650 GOSUB 710                                  <177>
660 FOR L=1 TO 5                               <031>
670 NEXT L                                     <110>
680 IF K<1000 THEN 630                         <052>
690 B6=TI                                      <123>
700 GOTO 730                                   <225>
710 RETURN                                     <086>
720 REM--------------------7---------          <177>
725 DIM M(10)                                  <092>
730 K=0                                        <009>
740 K=K+1                                      <009>
750 A=K/2*3+4-5                                <170>
760 GOSUB 830                                  <034>
770 FOR L=1 TO 5                               <142>
780 M(L)=A                                     <236>
790 NEXT L                                     <231>
800 IF K<1000 THEN 740                         <143>
810 B7=TI                                      <245>
820 GOTO 850                                   <093>
830 RETURN                                     <207>
840 REM--------------------8--------           <043>
850 K=0                                        <130>
860 K=K+1                                      <130>
870 A=K↑2                                      <135>
880 B=LOG(K)                                   <191>
890 C=SIN(K)                                   <205>
900 IF K<1000 THEN 860                         <022>
910 B8=TI                                      <090>
920 RETURN                                     <041>
930 REM -----------------------------          <077>
940 REM -         UP SORTIEREN      -          <248>
950 REM -----------------------------          <097>
960 REM -    Q1   = ANZAHL ELEMENTE -          <120>
970 REM -    FF$() = SORTIERFELD    -          <055>
980 REM -----------------------------          <127>
990 :                                          <027>
1000 REM JJ,LL = LAUFVARIABLE                  <125>
1010 REM Q2$   = ZWISCHENSPEICHER              <038>
1020 :                                         <057>
1030 REM SORTIERT DAS FELD FF$ MIT             <162>
1040 REM Q1 ELEMENTEN IN ALPHABE -             <115>
1050 REM TISCHER FOLGE                         <044>
1060 :                                         <098>
1070 FOR JJ=1 TO Q1-1                          <044>
1080 FOR LL=JJ+1 TO Q1                         <156>
1090 IF FF$(JJ)<=FF$(LL)THEN 1130              <208>
1100 Q2$=FF$(JJ)                               <062>
1110 FF$(JJ)=FF$(LL)                           <058>
1120 FF$(LL)=Q2$                               <086>
1130 NEXT LL                                   <136>
1140 NEXT JJ                                   <142>
1150 RETURN                                    <016>
```

Listing 1. Die in Tabelle 2 zusammengefaßten Ergebnisse wurden mit diesem Programm ermittelt. Die Zeit für zum Beispiel 1000 GOSUB-RETURN erhält man, indem die Zeit für Benchmark 4 von der Zeit für Benchmark 5 abgezogen wird.

schachtelte MID$-Statements, die man aber generell bei der Arbeit mit Compilern vermeiden sollte.

Was angenehm überrascht, ist die Tatsache, daß Austro-Speed sogar die variable Dimensionierung (zum Beispiel DIM A (B), zuläßt. Eine eigentlich gar nicht selbstverständliche Eigenschaft, da Compiler auf das feste Anlegen von Variablenfeldern angewiesen sind und somit deren Ausmaße beim Compilieren feststehen müssen.

Doch nun zum Petspeed, dessen Lieferumfang aus drei Blättern Druckerpapier und einer Diskette bestand. Das »3-Blatt-Handbuch« strotzt nur so von Fehlern und macht einen gleich einmal auf eine nette Überraschung gefaßt. Das Directory der beigefügten Diskette ist nur über Spezialprogramme zu listen. Dieser (unnötige) »Scherz« hätte unter normalen Umständen sicher nichts ausgemacht. Das Sonderbare an Petspeed ist nur, daß man sein Basic-Programm auf die Systemdiskette kopieren muß, damit der Compiler arbeiten kann. Aus diesem Grund hat mich dieser »Gag« ziemlich verärgert, da er die Möglichkeit eines Bedienungsfehlers geradezu herausfordert.

Hat man auch hier EDDI in die compilierfähige Form gebracht, so kann es losgehen. Der Petspeed ist ein 4-Pass-Compiler, was schon einmal gewisse Erwartungen bezüglich der Leistung weckt.

Nach 7 ½ Minuten ist die Arbeit an EDDI abgeschlossen, und wir erhalten, wie schon bei Austro-Speed, zwei Programm-Files zurück. Eines der beiden Programme ist dabei wieder eine Korrekturerleichterung, die alle vorhandenen Zeilennummern mit deren neuen Adressen enthält.

Der Petspeed-Compiler

Petspeed ist also beim Compilieren hinter Austro-Speed der zweitschnellste Compiler dieses Tests. Der einzige Nachteil besteht in der Tatsache, daß das zu compilierende Programm auf die Systemdiskette kopiert werden muß. Erstens ist damit die Wahrscheinlichkeit einer Panne mit der Originaldiskette größer, und zweitens wird der Platz zum Compilieren ganz erheblich eingeschränkt, da Petspeed schon über 200 Blöcke für sich beansprucht. So darf das zu übersetzende Basic-Programm auch nicht länger als 80 Blocks sein.

Sieht man sich die Gesamtlänge der vier Compilate aller Compiler an, so erkennt man, daß sich BASS-, Exbasic Level II-Compiler und Austro-Speed in etwa entsprechen. Petspeed hat mit Abstand das längste File hinterlassen, was sich auch bei der weiteren Arbeit mit diesem Compiler immer wieder zeigen wird.

Sie werden jetzt natürlich gespannt auf die Ergebnisse des Compilierens sein. Was ist eigentlich aus dem einstmals so langsamen Basic-Programm geworden?

Nun, ich will Sie nicht länger auf die Folter spannen. Allerdings habe ich zum Zeitvergleich nicht EDDI herangezogen, obwohl sich das Ergebnis (bei rückblickender Betrachtung) nicht verändert hätte. Es wurde ein Programm erstellt, in dem systematisch ein paar Befehlsgruppen abgefragt wurden, um die Geschwindigkeit in verschiedenen Bereichen vergleichen zu können.

Das Ergebnis der Messungen sehen Sie in der Zusammenfassung in Tabelle 1 und 2. Es zeigt sich ganz deutlich, daß Petspeed (obwohl der »Oldtimer« dieses Tests) der eigentliche Testsieger ist. Er hat in fast allen Bereichen die Nase vorne und erreicht Geschwindigkeiten, von denen seine Konkurrenten nur träumen können.

Unser Testprogramm absolvierte er beispielsweise in 145 Sekunden. Das Original unter dem Interpreter benötigt noch 391 Sekunden. Mit großem Abstand folgt dann erst einmal das Compilat von Austro-Speed. Es erreicht immerhin eine Zeit von 173 Sekunden und ist damit um fast 20 Prozent langsamer als Petspeed.

Enttäuscht hat in diesem Test der Exbasic Level II-Compiler. Einmal davon abgesehen, daß er schon eine Reihe anderer Schwächen aufzuweisen hatte, bildete er noch zusätzlich in diesem Geschwindigkeitstest das Schlußlicht mit einer Zeit von 198 Sekunden.

Der BASS-Compiler ist zwar genauso schnell oder langsam wie der Exbasic-Compiler (198 Sekunden), er hat aber immer noch den Vorteil seines positiven Konzepts, des bedienerfreundlichen Compilats, was zumindest Maschinensprach-Spezialisten zu schätzen wissen dürften.

BASS überholt Austro-Speed lediglich bei der Übersetzung von POKEs und PEEKs. Dieser Unterschied ist jedoch sehr gering und kann an dem Gesamtergebnis nichts ändern.

Fazit

Fangen wir mit dem Exbasic Level II-Compiler an. Dieses Produkt konnte in wesentlichen Punkten nicht überzeugen. Einige dieser Gründe sind mit Sicherheit darin zu sehen, daß dieser Compiler nur die um einen Assembler erweiterte Version des BASS ist, wobei jedoch die relativ schwache Leistung des BASS noch durch die verlorengegangenen positiven Eigenschaften dieses Programms verstärkt wurde. Der BASS ist zwar langsam, aber sein Vorteil liegt im offenen Konzept des erzeugten Codes. Damit hat der Benutzer die Möglichkeit, auch beim Compilat noch leicht Änderungen vorzunehmen. Zu diesem Zweck ist BASS sehr umfassend dokumentiert, im Gegensatz zum Exbasic Level II-Compiler.

Die Nachteile beider Compiler waren aber die »pingeligen« Ansprüche an die Syntax des zu compilierenden Programms. Ein Anwender, der mit Compilern arbeitet, möchte in der Regel ein älteres Programm ohne große Änderungen am Original schnell compilieren können. Das ist aber ohne große Änderungen bei diesen beiden Compilern fast nicht möglich.

Als Kaufempfehlung kann hier also höchstens BASS gelten (198 Mark). Dieser Compiler ist jedoch nichts für Anfänger und für Anwender, die sich einen Compiler nur zum bequemen »Hochpuschen« von Basic-Programmen zulegen. Der Exbasic Level II-Compiler ist allein schon seines hohen Preises wegen (298 Mark) gemessen an den Leistungen, nicht ohne Vorbehalt zu empfehlen. Er besitzt keine herausragenden Vorzüge und könnte bei weniger erfahrenen Programmierern schnell ein falsches Bild von Compilern hervorrufen.

Das Nachladen von Programmen, sogenanntes Overlay, machte bei allen Compilern dieses Tests (infolge der Variablenorganisation) Schwierigkeiten. Das im Interpretermodus mögliche Nachladen von Programmen mit Variablenübergabe (Warm-Overlay) ist in der Regel mit den Compilern nicht möglich. Hier muß man normalerweise eine Speicherstelle als Flag (beim Nachladen mehrere Programme) benutzen, da die Variablen durch das Nachladen gelöscht werden (Kalt-Overlay).

Am großzügigsten bei der Analyse von Programmen zeigte sich der Austro-Speed. Er war beim Compilieren mit Abstand der Schnellste. Die Endgeschwindigkeit des Compilats ist zwar nicht mit der des Petspeed vergleichbar; es zeigt sich jedoch deutlich, daß dieser Compiler die wenigsten Probleme aufwirft, zumal seine Bedienung ein wahres Kinderspiel ist. Er »verdaut« die meisten Programme ohne Schwierigkeiten und zeigt sich auch in der Anwendung sehr vielseitig, da er sowohl mit einem, als auch mit mehreren Floppy-Laufwerken zusammenarbeitet. Der Preis vom Austro-Speed ist mit dem von Exbasic Level II-Compiler identisch (298 Mark); hätte man also die Wahl zwischen beiden, so dürfte die Entscheidung nicht allzu schwer fallen.

Sehr viel fürs Geld bekommt man mit Petspeed ins Haus geschickt (149 Mark). Dieser Compiler übertraf alle Erwartungen und zeigte sich auch in der Bedienung recht einfach. Die zwei negativen Aspekte dürften hier wohl die unzureichende Literatur und das notwendige Kopieren des Basic-Programms auf die Systemdiskette sein. Wenn man aber einmal die 149 Mark sieht, die der Petspeed kostet, so zeigt sich dennoch ein hervorragendes Preis/Leistungsverhältnis, das von keinem Compiler des Tests erreicht wurde. In den Tabellen finden Sie zur besseren Orientierung noch einmal alle Test-Ergebnisse zusammengefaßt.

(Karsten Schramm/gk)

Bezugsadressen und Info:
BASS: gmbsoft, Kaiser-Friedrich Ring 55, 6200 Wiesbaden
Austro-Speed (Austro-Comp): Commodore, Lyoner Str. 38, 6000 Frankfurt/Main und für Österreich: Digimat, Arbeitergasse 48, 1050 Wien
Ex-Basic-Compiler II: Interface Age, Vohburger Str. 1, 8000 München 21
Petspeed: Infotronik, Birkenstr. 40, 4100 Duisburg

Geschwindigkeit durch Maschinencode – so arbeiten Compiler

Programme komfortabel in Basic schreiben, aber mit der Geschwindigkeit von Maschinensprache ausführen lassen — Compiler machen's möglich.

Es ist schon erstaunlich. Da gibt es Programme, die andere Programme als Eingabedaten verwenden und diese in reinen Maschinencode übersetzen, der von der CPU direkt ausgeführt werden kann. Das ist durchaus von Vorteil:

Basic-Programme sind nämlich einfach zu schreiben, aber langsam in der Ausführung. Maschinensprache ist dagegen sehr schnell, aber schwierig zu programmieren. Compiler bilden praktisch die Brücke zwischen den beiden grundsätzlichen Anforderungen nach einfacher Programmerstellung und hoher Ausführungsgeschwindigkeit.

Auf den ersten Blick drängt sich der Vergleich mit einem Assembler auf, der ja auch in Klartext gegebene Befehle in Maschinencode übersetzt. Doch der Vergleich hinkt, denn die Aufgabe eines Assemblers ist vergleichsweise trivial. Zu jedem Klartextbefehl (Mnemonic) wie zum Beispiel LDA gibt es nämlich genau einen Opcode. Der Assembler macht daher im wesentlichen nichts anderes, als in einer zweispaltigen Tabelle das Mnemonic zu suchen und bei Erfolg aus der zweiten Spalte den zugehörigen Opcode zu entnehmen. Zwischen Assembler-Mnemonic und erzeugtem Maschinencode besteht also ein Verhältnis eins zu eins.

Bei sehr maschinennahen Befehlen wie beispielsweise beim Basic-Befehl GOTO hat es der Compiler ähnlich einfach: »GOTO (Zeilennummer)« würde von einem 6502-Basic-Compiler übersetzt in »JMP (Adresse)«. Bei Befehlen wie »IF X = A + 3 THEN Z = 5« wird die Sache schon etwas komplizierter. Eine solche Anweisung kann allein schon deshalb nicht direkt übersetzt werden, weil der 6502-Prozessor keinen Maschinenbefehl »IF« kennt.

Hier muß der Compiler also wesentlich mehr leisten, als nur Opcodes zu bestimmten Schlüsselwörtern herauszusuchen. In der Regel entspricht einem Befehl in einer höheren Programmiersprache eine ganze Befehlssequenz auf der Maschinenebene. Die Aufgabe eines Compilers wird daher mit Recht als Übersetzung statt Assemblierung (Zusammenfügung) bezeichnet.

Übersetzt wird immer aus einer höheren Programmiersprache (beispielsweise Basic) in eine maschinennahe »Sprache«, meistens direkt in Maschinencode. Einer Programmiersprache liegt — wie auch jeder natürlichen Sprache — ein Vokabular und eine Grammatik zugrunde.

Auf die Grammatik kommt es an

Vokabeln der Sprache Basic sind zum Beispiel GOTO, PRINT, DATA, aber auch Zahlen und Variablennamen gehören dazu. Daneben gibt es bestimmte »Satzzeichen« wie » = «, »:« etc.

Eine Grammatik ist ja nichts anderes als eine Menge von Regeln, die angeben, wie aus den zur Verfügung stehenden Zeichen und Wörtern korrekte »Sätze« gebildet werden. Folgende drei Regeln bilden zum Beispiel eine einfache Grammatik:

Regel 1: Ein Satz besteht aus einem Subjekt, gefolgt von einem Prädikat.

Regel 2: Ein Subjekt ist eines der Worte HUNDE, COMPUTER, MENSCHEN.

Regel 3: Ein Prädikat ist eines der Worte LESEN, RECHNEN, BELLEN, SCHLAFEN.

In der sogenannten Backus-Naur-Form, die zur Definition der Programmiersprache Algol entwickelt wurde, lassen sich die drei Regeln kürzer und eindeutiger darstellen:

1. ⟨Satz⟩ ::= ⟨Subjekt⟩ ⟨Prädikat⟩

Befehl	Bedeutung
LIT n	Konstante n auf Stack legen
RCL x	Inhalt der Variablen x auf Stack legen
STO x	obersten Stapelwert nach x speichern
ADD	die obersten beiden Stapelwerte addieren
SUB	die obersten beiden Stapelwerte subtrahieren
MUL	die obersten beiden Stapelwerte multiplizieren
DIV	die obersten beiden Stapelwerte dividieren
JMP a	unbedingter Sprung zur Adresse a
JEQ a	Sprung, falls oberstes Stapelelement Null
JNE a	Sprung, falls oberstes Stapelelement nicht Null
JGE a	Sprung, falls oberstes Stapelelement positiv
JLT a	Sprung, falls oberstes Stapelelement negativ
JSR a	Unterprogrammaufruf
RTS	Rückkehr vom Unterprogramm
SQR	Quadratwurzel aus oberstem Stapelelement
SIN	Sinus des obersten Stapelelements
COS	Cosinus des obersten Stapelelements
TAN	Tangens des obersten Stapelelements
ATN	Arcustangens des obersten Stapelelements
LOG	Logarithmus des obersten Stapelelements
EXP	Exponentialfunktion des obersten Stapelelements

Tabelle 1. Typische Befehle eines einfachen Zwischencodes

Basic-Befehl	erzeugter Zwischencode
A = 3 + 5*X	LIT 3, LIT 5, RCL X, MUL, ADD, STO A
GOTO 123	JMP xxxx
GOSUB 500	JSR xxxx
RETURN	RTS
IF A = 0 THEN 50	RCL A, JEQ xxxx

Tabelle 2. Beispiele zur Übersetzung von Basic in Zwischencode

2. ⟨Subjekt⟩ ::= HUNDE | COMPUTER | MENSCHEN
3. ⟨Prädikat⟩ ::= LESEN | RECHNEN | BELLEN | SCHLAFEN

In spitzen Klammern stehen dabei immer Verweise auf andere Regeln, die senkrechten Striche trennen verschiedene Alternativen.

Mit dieser Grammatik können durch einfache Anwendung der Regeln Sätze erzeugt werden wie »HUNDE BELLEN« oder »MENSCHEN LESEN« und einige mehr.

Nach Regeln dieser Art ist auch die Basic-Grammatik aufgebaut. Um alle syntaktisch korrekten Sprungbefehle zu beschreiben, genügen folgende Regeln:

1. ⟨Sprungbefehl⟩ ::= GOTO ⟨Zeilennummer⟩
2. ⟨Zeilennummer⟩ ::= {⟨Ziffer⟩}
3. ⟨Ziffer⟩ ::= 0|1|2|3|4|5|6|7|8|9

Regel 1 bestimmt, daß ein Sprungbefehl aus dem Schlüsselwort GOTO, gefolgt von einer Zeilennummer, besteht. Regel 2 besagt, daß eine Zeilennummer aus einer Folge von Ziffern besteht (die geschweiften Klammern deuten die Wiederholung an). Regel 3 schließlich definiert, was eine Ziffer ist. Bleibt noch zu bemerken, daß eine Zeilennummer 999999 zum Beispiel syntaktisch richtig ist, aber aus anderen Gründen nicht zulässig ist.

Häufig werden anstelle der Backus-Naur-Form Syntax-Graphen als anschaulichere Art der Darstellung gewählt. Bild 1 zeigt den Anfang einer syntaktischen Definition des allgemeinen Begriffes »Basic-Programm«, und zwar die ersten drei Regeln. Die Diagramme werden grundsätzlich von links nach rechts gelesen, jede andere Richtung muß durch entsprechende Markierungen angegeben werden.

Regel 1 aus Bild 1 besagt, daß ein Basic-Programm zunächst aus einer Programmzeile besteht. Dann kann entweder bereits Schluß sein, oder aber eine weitere Programmzeile folgen und so fort. Regel 2 definiert den Begriff der Basic-Zeile, bestehend aus Zeilennummer und einer Anweisung, worauf ein Doppelpunkt und eine Anweisung etc. folgen können.

Was hat das Ganze nun mit Compilern zu tun? Sehr viel, denn eigentlich sind wir schon mittendrin im Thema. Jeder Compiler verfügt nämlich über einen sogenannten »Parser«, ein wichtiges Teilprogramm, das die syntaktische Analyse übernimmt. Dazu ist es notwendig, daß der Parser die Syntax genau »kennt«. Das sieht dann so aus, daß der Parser bei der Programmanalyse bildlich gesprochen den Linien der Syntax-Diagramme folgt. Stößt er dabei auf ein Kästchen mit einer Syntax-Regel, dann wird sofort das entsprechende Unterprogramm zur Analyse oder Übersetzung dieser Regel aufgerufen. Die Eingabe für den Parser ist also der Programm-Quelltext, als Ausgabe liefert er einen Zwischencode, also eine Art vor-übersetztes Programm.

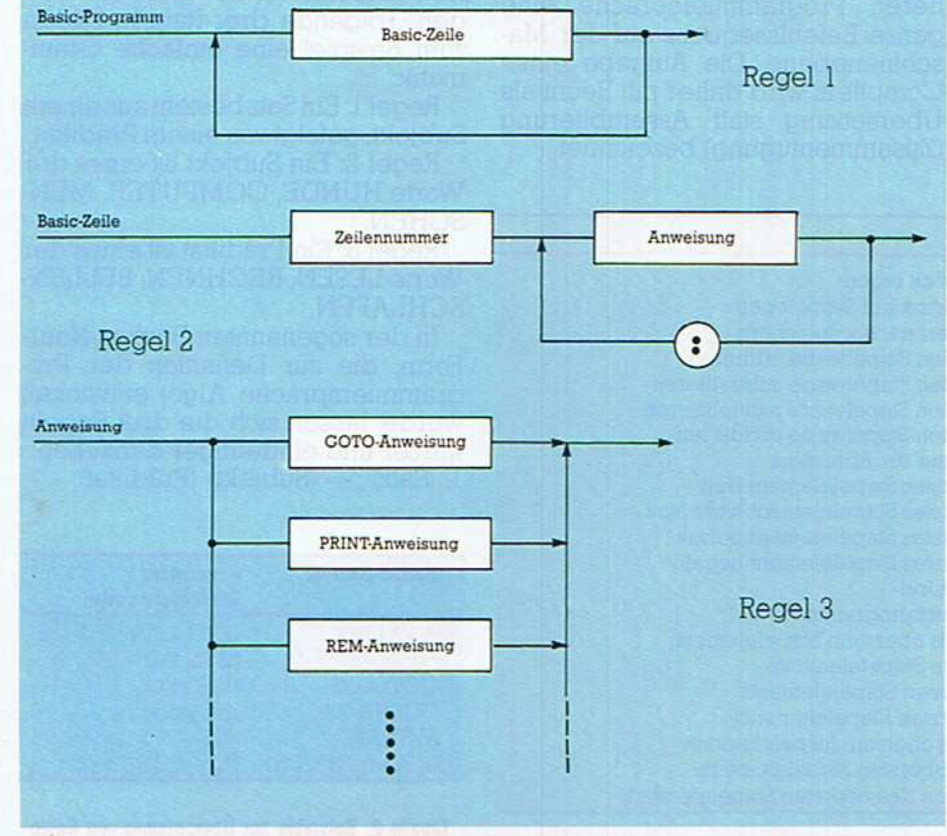


Bild 1. Syntax-Diagramme zur Definition des Begriffs »Basic-Programm«

Bei diesem Zwischencode handelt es sich in der Regel — man höre und staune — um den Maschinenbefehlssatz eines Computers, den es gar nicht gibt. Tabelle 1 zeigt, wie der Befehlssatz eines solchen hypothetischen Computers aussehen könnte. Natürlich erzeugt der Compiler intern nicht die ASCII-Zeichenfolge für diese Befehle, sondern legt jeden Befehl in einem Byte codiert ab. Ein solches oft als »Token« bezeichnetes Byte ist im folgenden immer gemeint, wenn von einem vom Compiler erzeugten Zwischencode die Rede ist.

Intern arbeitet der »Ghost-Computer«, dessen Befehlssatz gerade der Zwischencode ist, mit einem Stack, wie er jedem Forth-Programmierer, aber auch jedem Benutzer von Hewlett-Packard-Taschenrechnern vertraut sein dürfte. Diesen Rechenstack kann man sich bildlich als Papierstapel vorstellen. Während der Berechnung eines mehr oder minder komplizierten arithmetischen Ausdrucks können Zettel mit Zwischenergebnissen auf den Stapel gelegt oder aber von oben weggenommen werden.

Natürlich kann man mit den Befehlen aus Tabelle 1 arithmetische Ausdrücke nicht in der gewohnten Schreibweise berechnen, sondern es muß zuvor eine Umwandlung in die sogenannte »umgekehrte polnische Notation« (UPN) erfolgen. Ein Ausdruck wie »2 + 3 x 5« würde beispielsweise folgenden Zwischencode ergeben (Stackinhalt in Klammern):

```
LIT 2    (2)
LIT 3    (2,3)
RCL X    (2,3,5)
MUL      (2,15)
ADD      (17)
```

Diese Übersetzung in Zwischencode ist nun relativ einfach zu automatisieren, denn auch für arithmetische Ausdrücke gibt es eine Art Grammatik, die auch die »Punkt-vor-Strich« - Regel berücksichtigt (Bild 2). Der Compiler »kennt« diese Regeln und wendet sie an. Sobald er beispielsweise innerhalb einer Formel auf eine Zahl (Konstante) stößt, erzeugt er den Zwischencode »LIT ⟨Konstante⟩«, bei einer Variablen den Zwischencode »RCL ⟨Variablenadresse⟩« und so fort.

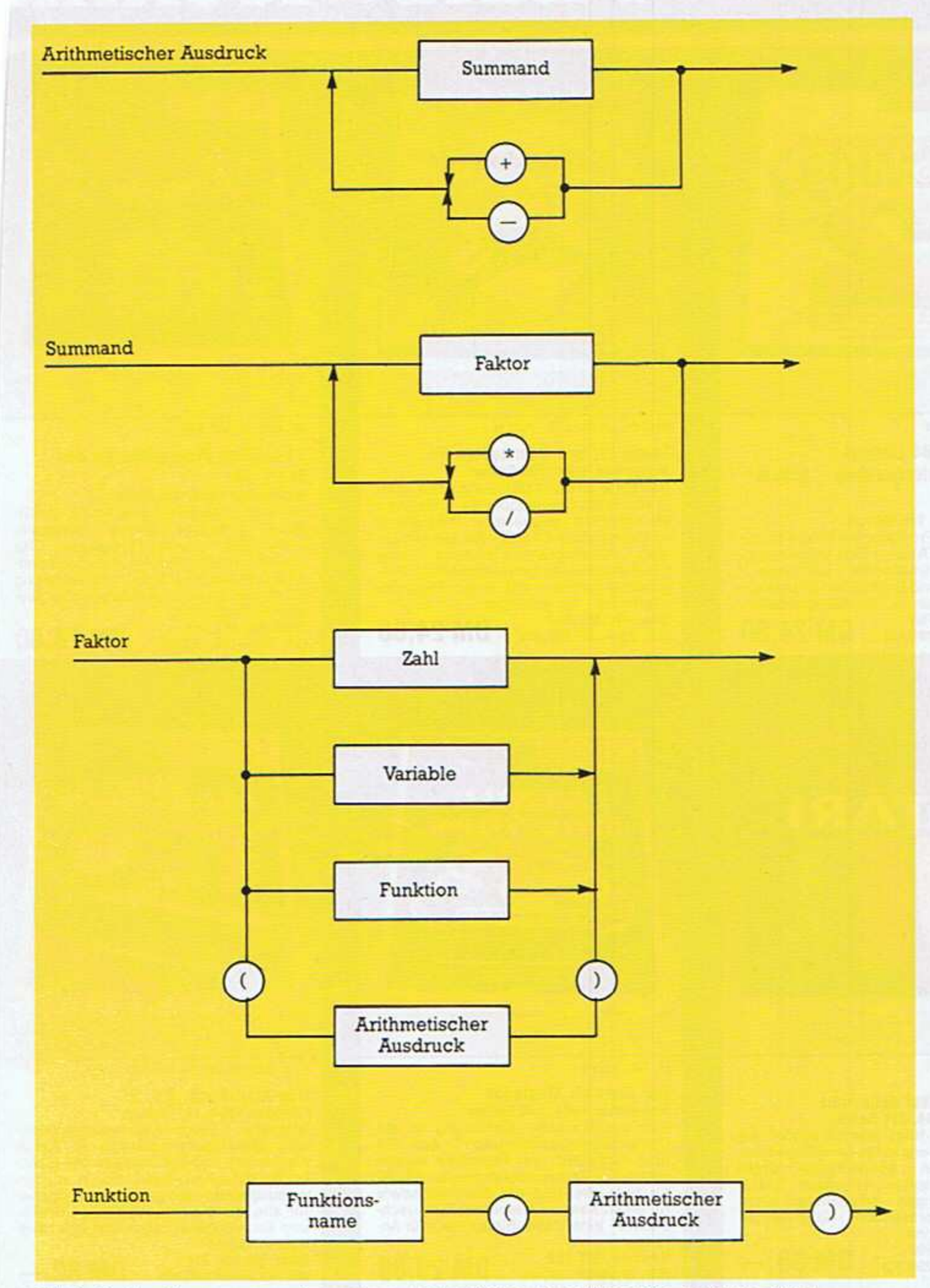


Bild 2. Syntax-Diagramme zur Auswertung arithmetischer Ausdrücke. Die »Punkt-vor-Strich«-Regel ergibt sich aus der Darstellung von selbst.

Tabelle 2 zeigt einige Beispiele für die Übersetzung von Basic-Anweisungen in Zwischencode. Natürlich ist der hier vorgestellte Zwischencode nur ein (unvollständiges) Beispiel. Sie sollten daher nicht erwarten, daß Ihr Basic-Compiler genau diesen speziellen Code erzeugt.

Ist Ihnen bis hierhin etwas aufgefallen ? In der bisherigen Beschreibung der Funktionsweise eines Compilers wurde weder darauf eingegangen, in welcher Sprache der Compiler selbst geschrieben ist, und vor allem nicht, für welchen Prozessor die Übersetzung durchgeführt wird. Der Grund dafür ist einfach: Ein Compiler ist ein Programm, das einem bestimmten Algorithmus realisiert. Es ist also völlig gleichgültig, in welcher Programmiersprache der Compiler geschrieben wurde. So kann ein Pascal-Compiler durchaus in Basic geschrieben werden, wenn das aus verschiedenen Gründen auch nicht besonders sinnvoll wäre. Außerdem haben wir bisher auch noch keinen wirklichen Maschinencode erzeugt, es ist also egal, für welchen Prozessor die Übersetzung stattfinden soll.

Einige Compiler belassen es sogar bei diesem Zwischencode. Es wird dann ein spezielles Interpreterprogramm benötigt, das diesen Zwischencode interpretiert (»Run-Time-Modul«). Derartige Compiler sind besonders bei 8-Bit-Computern verbreitet, da der Zwischencode sehr kompakt ist (ein Byte pro Befehl). Nachteilig ist natürlich, daß dieser Code immer noch interpretiert werden muß, wenngleich das wegen der sehr einfachen Struktur recht schnell geht. Derartige Zwischencode-Programme sind in der Ausführungszeit etwa zwei- bis zehnmal so schnell wie Basic, bei vermindertem Platzbedarf. Der sogenannte »P-Code«, den Pascal-Compiler erzeugen, ist übrigens ebenfalls ein solcher Zwischencode.

Die Erzeugung von Maschinencode

Die Erzeugung von reiner Maschinensprache aus dem Zwischencode ist nun relativ einfach und funktioniert ähnlich wie bei einem Assembler. Statt des Zwischencodebefehls ADD wird zum Beispiel bei einem 6502-System die Befehlsfolge für eine 16-Bit-Addition oder auch einfach nur der Code für »JSR ADD« erzeugt. Dadurch, daß der so erzeugte Code vom Prozessor direkt ausgeführt werden kann, ergeben sich sehr günstige Ausführungszeiten. Solcherart compilierte Programme laufen zwischen zehn- und hundertmal schneller als über einen Interpreter. Allerdings ist der Speicherbedarf gegenüber Zwischencodeprogrammen in der Regel um etwa das Doppelte erhöht.

Interessant ist, daß die unterschiedlichen Eigenschaften der verschiedenen Prozessoren erst bei der Erzeugung von Maschinencode aus dem Zwischencode zum Tragen kommen. Außerdem hatten wir festgestellt, daß es egal ist, in welcher Sprache ein Compiler geschrieben wird. Diese beiden Erkenntnisse haben weitreichende Konsequenzen bei der Entwicklung von Compilern.

Bootstrapping

Eine in der Praxis fast immer angewandte Methode der Compilerentwicklung beruht darauf, den Compiler in der Programmiersprache zu schreiben, die er übersetzen soll. Ein Basic-Compiler würde daher selbst in Basic geschrieben werden.

Der Grund dafür ist einfach: Hat man einmal einen (wenn auch noch sehr einfachen) in Basic geschriebenen Basic-Compiler zur Verfügung, dann kann dieser Compiler sich

Fortsetzung auf Seite 163

Impossible Mission



Wie unmöglich die Aufgabe ist, die man zu erfüllen hat, muß jeder Spieler für sich selbst herausfinden. Schnell langweilig wird dieses Spiel bestimmt nicht.

Der berüchtigte »Elvin« droht die Weltbevölkerung mit Atomwaffen zu vernichten. Als Mitglied der Anti-Computer-Terroristen-Gruppe sollen Sie »Elvin« ausschalten. Sie müssen einen Weg durch die Räume und Tunnel seines unterirdischen Hauptquartiers finden und dabei die Roboterwachen des wahnsinnigen Verbrechers überlisten. Sechs Stunden stehen ihnen zur Verfügung. Aber Vorsicht! Mit jedem »Ausfall« der Spielfigur verlieren Sie zehn Minuten. Die Zeit wird schnell knapp.

Suchen Sie nach Hinweisen, um das Paßwort zu bekommen, mit dem Sie bis zu Elvins Kontrollraum gelangen. Sie müssen schneller als die Wachen sein, sie überspringen oder sie mit gefundenen Paßworten für kurze Zeit außer Kraft setzen.

Bei jedem neuen Versuch werden der Aufbau der Räume und die Fähigkeiten der Roboterwachen geändert. War ein Raum beim letzten Versuch noch leicht zu bewältigen, ist er vielleicht dieses Mal kaum noch zu durchqueren.

»Impossible Mission« verfügt über eine eingebaute Sprachausgabe. Gleich am Anfang werden Sie von der Stimme »Elvins« begrüßt. Zu diesem Zeitpunkt scheint sich »Elvin« noch keine allzugroßen Sorgen über Ihr erscheinen zu machen, denn er ist der Meinung, Sie würden Ihr Leben verlieren.

In den einzelnen Räumen müssen Sie versuchen, Puzzle-Teile zu finden. Insgesamt sind 36 Teile versteckt. Jeweils vier Teile ergeben eine Lochkarte, auf der immer ein Buchstabe des Paßwortes für den Kontrollraum abgespeichert ist. An einigen Stellen finden Sie »Hilfspaßworte«. Diese können zwei Funktionen haben: Die einen dienen dazu, die Liftplattformen nach einer Veränderung in ihren Ausgangszustand zu versetzen. Die anderen machen alle Roboterwachen in einem Raum für kurze Zeit funktionsuntüchtig. Diese Zeit reicht meist aus, um zwei Gegenstände in einem Raum zu untersuchen.

Es gibt noch eine andere Möglichkeit, um an »Hilfspaßworte« zu kommen. In dem Labyrinth gibt es zwei Computerräume, in denen eine Konsole und eine Anzeigetafel untergebracht sind. Stehen Sie vor der Konsole und bewegen den Joystick nach vorne, erklingen drei Töne, zu denen jeweils ein Feld der Anzeigentafel aufleuchtet. Danach erscheint eine mittels Joystick zu steuernde Hand, mit der Sie die Felder anfahren können. Ihre Aufgabe besteht darin, die Töne vom tiefsten bis zum höchsten nachzuspielen. Dazu müssen Sie die Hand auf das jeweilige Feld positionieren und den Feuerknopf drücken. Haben Sie die richtige Reihenfolge gefunden wird Ihnen ein »Hilfspaßwort« gutgeschrieben, und die Anzahl der nachzuspielenden Töne erhöht sich um Eins. Mit ein wenig Geschick können Sie einige Hilfsmittel dazu bekommen.

Hervorragend ist auch die Animation dieses Spieles. Der Spielablauf gleicht einem Zeichentrickfilm, nur mit dem Unterschied, daß Sie den Ablauf steuern.

Liebhaber von Labyrinth- und Geschicklichkeitsspielen werden von Impossible Mission sicherlich begeistert sein. Die schwierigen Bilder und der sich stets ändernde Aufbau verhindern Langeweile.

Ab Mitte Januar wird Impossible Mission (Epyx) auch in Deutschland erhältlich sein. Der Preis stand bei Redaktionsschluß noch nicht fest.

(rg)



Abenteuer in der Höhle

Gordon Saga ist ein deutsches Abenteuerspiel der Spitzenklasse. Selbst einen Vergleich mit »Hobbit« oder ähnlichen Spielen muß es nicht scheuen.

Begeben Sie sich in die Welt der großen Abenteuer. Schlüpfen Sie in die Rolle eines Höhlenforschers. Gordon Saga ist der erste, in sich abgeschlossene Teil einer deutschen Abenteuerspiel-Serie. Am Ende dieses Teils wird Ihnen dann die »große Aufgabe« gestellt, auf die die folgenden Teile aufbauen.

Alle Textein- und ausgaben erfolgen in Deutsch. Allein dieser Umstand hebt dieses Spiel von vielen anderen, die meist in Englisch geschrieben sind, ab. Das Spiel akzeptiert sogar ganze Sätze. So könnte eine Anweisung zum Beispiel lauten: »NIMM DAS SCHWERT UND GEH NACH NORDEN«. Wird eine Eingabe trotz des recht großen Wortschatzes nicht verstanden, so meldet dies der Computer. Anweisungen, die er versteht, im Moment aber sinnlos sind, ignoriert er. Man kann allerdings, sofern man Lust dazu hat, zum Beispiel singen, beten oder warten, ohne daß dies den Spielablauf beeinflußt.

Einer der größten Vorzüge dieses Abenteuerspiels ist der variable Spielverlauf. Wie auch beim »Hobbit« muß man auf Überraschungen gefaßt sein. Überall lauern Gefahren, die Hilfsmittel sind nicht immer an den gleichen Stellen zu finden. Jeder neue Versuch, das Geheimnis zu ergründen, konfrontiert Sie mit neuen Bedingungen.

Spielspaß zum Sparpreis

In manchen Situationen wird es besonders schwierig. Sie geraten unter Zeitdruck. So fällt eine Tür hinter Ihnen zu und die Zimmerdecke senkt sich oder es rollt eine große Steinkugel auf Sie zu. Hier muß der Abenteurer schnell und richtig reagieren, um sein Leben zu behalten. Aber die tödlichen Unfälle halten sich in Grenzen. Doch kann es leicht vorkommen, daß ein Monster gerade das nötigste Hilfsmittel stiehlt und irgendwo versteckt.

Ein niedriger Preis muß nicht immer ein Hinweis auf schlechte Qualität sein. So macht sich auf dem deutschen Softwaremarkt ein neuer Trend bemerkbar. Qualitativ gute Software wird immer häufiger zu einem fairen Preis angeboten. So können sich zum einen auch C 64-Besitzer, die nicht über große finanzielle Mittel verfügen, gute Programme leisten. Zum anderen wird den Raubkopierern das Hauptargument (der hohe Software-Preis) genommen. Auch dieses Abenteuerspiel kommt mit einem Preis von 39 Mark diesem Trend nach. (rg)

Info: Markt und Technik Verlag AG, Happy Software, Hans-Pinsel-Str. 2, 8013 Haar bei München
Preis: 39 Mark

Die Lösung von Hobbit

Lange haben die Abenteuerspielfreunde auf die Lösung von Hobbit warten müssen. Einen der möglichen Lösungswege, der sogar ein Ergebnis von 101,5% zuläßt, wollen wir Ihnen hier vorstellen.

The Hobbit ist ein Grafikadventure, bei dem es darauf ankommt einen Schatz zu finden.

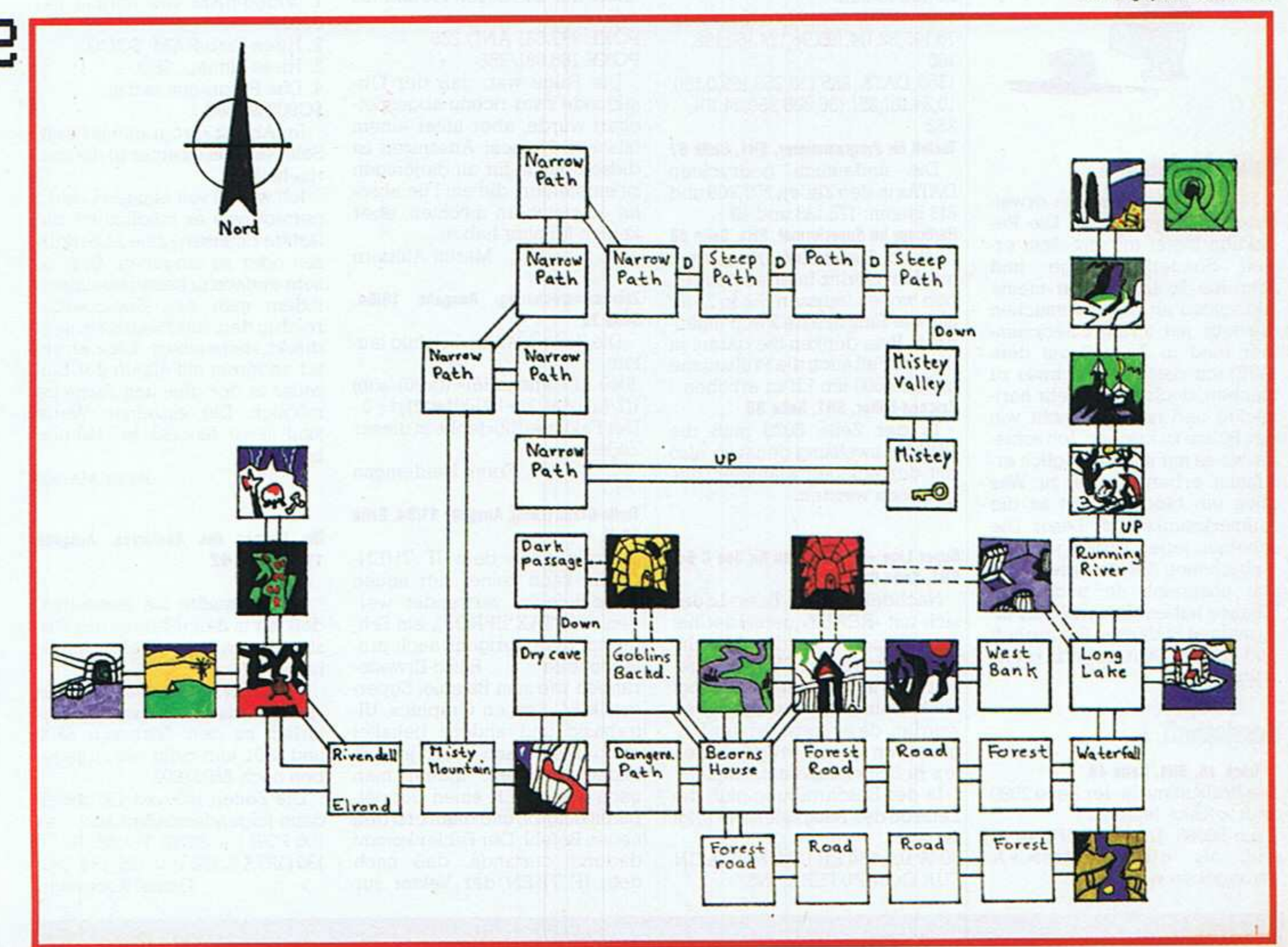
Eine Standard-Lösung dazu ist nicht möglich, da sich durch umherirrende Gestalten, ständig neue Spielsituationen ergeben.

Befehlsliste:

Die in Klammern angegebenen Hinweise brauchen nicht mit eingegeben zu werden. Richtungsangaben, wie zum Beispiel »East«, können durch die Anfangsbuchstaben abgekürzt werden.

START:
OPEN DOOR, EAST, NORTH, NORTH, WAIT (bis »DAY DAWNS«), SOUTH, GET KEY, NORTH, UNLOCK, OPEN, GO DOOR (Rocky door), GET ROPE, GET SWORD, SOUTH, SOUTH, SOUTHEAST, TALK, SAY ELROND »READ MAP« (bis er sie liest), EAST, NORTH, NORTHEAST, NORTH, SOUTHEAST, DOWN, DOWN, DOWN, DOWN, EAST, GET KEY, UP, NORTH, WEST, SOUTH, EAST, NORTH, (nun warten Sie, bis ein Goblin Sie gefangen nimmt, achten Sie aber darauf, daß Gandalf oder Thorin bei ihnen sind), DIG SAND (in »Goblins dungeon«), BREAK TRAP WITH SWORD, GET KEY, SAY THORIN »CARRY ME« (oder zu Gandalf), SAY THORIN »OPEN WINDOW«, SAY THORIN »GO WINDOW«, SOUTHWEST, (als nächstes müssen Sie in dem Verlies der Goblins einen Ring suchen, der von Gollum bewacht wird. Haben Sie diesen, begeben sie sich zum Ausgang »GOBLINS BACKDOOR«), EAST, EAST, OPEN CURTAIN, OPEN CUPBOARD, GET, NORTHEAST, EAST, EAST, WEST, WEST, WAIT (bis »Wooden Elf« kommt und Sie in einem roten Kerker einsperrt) (Sie befinden sich nun im Kerker), WAIT (bis »SOMEONE OPEN RED DOOR«), GO DOOR, NORTH, WAIT,

Der Lageplan von Hobbit



SOUTH, (diese »NORTH-WAIT-SOUTH«-Kombination so lange, bis »BUTLER DRINK SOME WINE«, dann weiter bis »BUTLER OPEN TRAP DOOR«, wenn »BUTLER THROWS BARREL THROUGH TRAP DOOR«, »JUMP BARREL« eingeben), EAST, SAY BARD »CAREFULLY, GO NORTH«, NORTH, NORTH, WAIT (bis »RED DRAGON« erscheint), SAY BARD »SHOOT DRAGON«, UP, NORTH, NORTH, NORTH, GET TREASURE, SOUTH, SOUTH, SOUTH, DOWN, SOUTH, SOUTH, (am »WATERFALL« warten Sie bis Wooden-Elf kommt) so kommt man in das Elfenverlies), WAIT (auf »SOMEONE OPEN RED DOOR«), GO DOOR, NORTH, WAIT, WEAR RING, READ MAGIC DOOR, LOOK DOOR, WAIT (irgendwann öffnet sich das magische Tor), WEST, (ein Spinnennetz behindert nun den Weg, dieses Hindernis kann durch Eingabe von SMASH WEB beseitigt werden), (nun versuchen Sie an Hand der Lagekarte zum LONELAND zurückzukehren), GO DOOR, OPEN CHEST, PUT TREASURE.
ENDE

Mit Hilfe dieser Lösung sollte es möglich sein mindestens 85% des Adventures zu lösen. Wenn man alle Räume durchlaufen hat, sind sogar 101,5% drin. Übrigens der Punktestand kann mit »SCORE« abgefragt werden.

Und nun viel Erfolg!

(Roland Selzer/rg)



Fehlerteufelchen

Nun habe ich also ein erweitertes Betätigungsfeld. Die Redaktion bietet mir mit dem ersten Sonderheft sage und schreibe 26 Listings für meine Aktivitäten an. Zwar versuchen die jetzt mit ihrem Checksummer (und in Zukunft mit dem MSE) mir das Leben schwer zu machen, doch ich bin sehr hartnäckig und nicht so leicht von der Bühne zu kriegen. Ich schlage, wo es mir immer möglich erscheint, erbarmungslos zu. Was mich ein bißchen stört ist die Aufmerksamkeit der Leser. Die scheinen jeden Artikel von vorne bis hinten durchzulesen. Wie soll unsereins da noch eine Chance haben? Ich muß hier also meinen Einfluß im Sonderheft und im 64'er Stammagazin preisgeben.

Sonderheft 1

Track 18, SH1, Seite 46

Die Prüfsumme in der Zeile 2520 muß <230> lauten.
Zeile 50005: Das geSHIFTete Pi muß als <Commodore> + A eingegeben werden.

Disksorter in Vollendung, SH1, Seite36

Zeile 3160: Auch hier muß das geSHIFTete Pi als <Commodore> + A eingegeben werden.

Disketten-Meister, SH1, Seite 51

Zwei Zeilen wurden nur halb abgedruckt. Die kompletten Zeilen lauten:

```
1020 DATA 134,193,76,174,167,32,
76,195,32,114,195,96,174,160,192,
160
1350 DATA 255,133,253,169,0,160,
10,24,101,251,136,208,250,24,101,
252
```

Toolkit für Programmierer, SH1, Seite 67

Die undeutlich gedruckten DATAs in den Zeilen 207,208 und 213 lauten: 173,133 und 19

Hardcopy im Superformat, SH1, Seite 86

Wenn Sie dieses Programm mit dem Görlitz-Interface betreiben wollen, müssen Sie in Zeile 320 die Eins in eine Zwölf umändern. Bitte denken Sie daran, in diesem Fall auch die Prüfsumme in Zeile 600 um Elf zu erhöhen.

Zeichen-Editor, SH1, Seite 88

In der Zeile 5020 muß die PRINT-Anweisung genauso, also mit der eckigen Klammer, eingegeben werden.

Super Line — 80 Zeichen für den C 64, SH1, Seite 91

Nachdem der Basic-Lader sich mit »READY« gemeldet hat (beziehungsweise das Maschinenprogramm mit SYS 36864 gestartet wurde), muß unbedingt noch einmal NEW eingegeben werden, da es sonst bei der Eingabe von neuen Programmzeilen zu Schwierigkeiten kommt.

In der Beschreibung muß die Zeile 30 des Beispieles wie folgt lauten:

30 W 0,0,"64'ER DAS MAGAZIN FÜR COMPUTER-FANS"

64'er, Ausgabe 10, 11, 12

Synthesizer (AdM), Ausgabe 12/84, Seite 55

Das Programm selbst ist in Ordnung, nur die SAVE-Routine weist einen kleinen Schönheitsfehler auf. Die Zeilen 140 und 150 müssen lauten:

```
POKE 187,681 AND 255
POKE 188,681/256
```

Die Folge war, daß der Objektcode zwar richtig abgespeichert wurde, aber unter einem falschen Namen! Ansonsten ist diese Routine für all diejenigen zu empfehlen, die ein File absolut abspeichern möchten, aber keinen Monitor haben.

Martin Ahlborn

Zinseszinsrechnung, Ausgabe 10/84, Seite 72

Die Zeile 3040 muß richtig lauten:

```
3040 iff=3thena(16)=((a(10)/a(9))
↑(1/(a11)*a(12)))-1)*100*a(12):d=0
```

Der Faktor »a(12)« fehlte in dieser Zeile.

Frank Heidemann

Turtle-Grafik (LdM), Ausgabe 11/84, Seite 55 ff.

Direkt nach dem IF...THEN-Befehl kann keiner der neuen Basic-Befehle verwendet werden (SYNTAX ERROR), ein Fehler mit dem übrigens auch professionelle Basic-Erweiterungen wie zum Beispiel Supergrafik 64, Screen Graphics, Ultrabasic und andere behaftet sind. Dies kann man jedoch leicht umgehen, indem man nach dem THEN einen Doppelpunkt eingibt, und dann erst den neuen Befehl. Der Fehler kommt dadurch zustande, daß nach dem IF...THEN der Vektor zur Ausführung eines Basic-Befehls übersprungen wird und so die neuen Befehle nicht ausgeführt werden können.

Die korrekte Speicheraufteilung von Turtle-Graphics lautet übrigens:

1. Video-RAM wie normal bei \$0400
2. Hires-Farb-RAM: \$CC00
3. Hires-Bitmap: \$E000
4. Das Programm selbst: \$C000-\$C88B

Im Absatz »Programmierung«, Seite 49, muß es unter b) »f5« statt »fs« heißen.

Ich wurde von einigen Lesern gefragt, wie es möglich ist, die lästige Einleseroutine zu verkürzen oder zu umgehen. Dies ist sehr einfach zu bewerkstelligen, indem man den Speicherbereich in dem das Programm liegt direkt abspeichert. Dies ist unter anderem mit einem der Einzeiler in der gleichen Ausgabe möglich. Die einzelnen Werte sind: ls = 0, hs = 192, le = 140 und he = 200.

Peter Menke

Die Ebenen des Absturzes, Ausgabe 11/84, Seite 92

Leider mußte ich feststellen, daß mir in den »Ebenen des Absturzes« ein kleiner Fehler unterlaufen ist.

Nach Erkennen von »cbm 80« springt das Betriebssystem natürlich zu den Adressen \$800 und \$801, und nicht wie angegeben nach \$802/803.

Die Zeilen 100 und 130 sehen dann folgendermaßen aus:

```
100 FOR I = 32768 TO 32778
130 DATA 9, 128, 0, 0, 195, 194, 205.
```

Daniel Kossmann

DAS GRAB DES PHARAO



Ich wurde am 31.1.1965 in Nürnberg geboren und wohne seit 1976 in Stein. An dem Nürnberger Dürer-Gymnasium absolviere ich die 13. Klasse. Meine Programmierkenntnisse erwarb ich jedoch nicht durch einen Informatikkurs, sondern durch Selbststudium mit einem VC 20, den ich Mitte 1983 erwarb. Da ich schon bald an die Grenzen dieses Systems stieß und mir überdies bewußt wurde, daß diesem Computer keine lange Zukunft mehr auf dem Markt beschert sein würde, entschloß ich mich Anfang 1984, auf den C 64 umzusteigen. Frustriert durch mehrere Versuche, Actionspiele in Basic zu programmieren, begann ich schon kurz darauf Maschinensprache zu lernen. Nachdem ich jedoch mit großer Begeisterung einige kommerzielle Adventures gespielt hatte, reifte in mir der Entschluß, selbst ein solches Spiel zu schreiben. Da ich überdies eine gewisse Faszination für das alte Ägypten mit seinen geheimnisvollen Pyramiden empfinde, lag das Thema für mein Programm auf der Hand. Das vorliegende Grafikadventure »Grab des Pharaos« ist das Ergebnis meiner Bemühungen.

(Wolfgang Rausch)

Ein Abenteuer im alten Ägypten erwartet Sie diesmal als Listing des Monats.

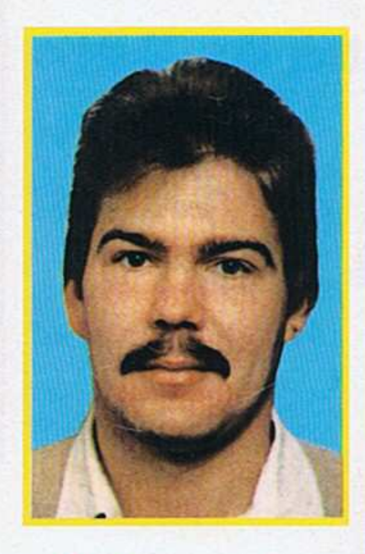
Ziel des Grafikadventures ist es, die goldene Totenmaske des Pharaos zu finden, wozu eine Pyramide erforscht werden muß, in der zahlreiche Fallen gegen Grabräuber verborgen sind. Am Anfang des Spiels befindet sich der Abenteurer jedoch noch mitten in der Wüste und muß die Pyramide erst einmal finden.

Das Programm besteht aus 58 Bildern, die sich innerhalb der Pyramide aus verschiedenen Komponenten (Durchgänge, Türen, Treppen und so weiter) zusammensetzen. Aufgrund der Vielzahl und Ähnlichkeit der Räume ergibt sich ein wahrer Irrgarten. Die Bilder werden, um den Effekt von hochauflösender Grafik zu simulieren, aus über 80 neudefinierten Zeichen und 33 Sprites zur Darstellung der Gegenstände aufgebaut. Die Befehlsauswertung der Eingaben folgt getrennt in zwei »INPUT« Statements: Das heißt nach jedem eingegebenen Wort muß »RETURN« gedrückt werden, wodurch sich jedoch kein Nachteil ergibt.

Das Programm versteht folgende 18 Verben: gehe, zerstöre, öffne, klopfe, frage, hacke, töte, krieche, werfe, schiebe, drehe, nimm, verliere, ziehe, list, save, stop.

Da großer Wert auf die aufwendige Grafik gelegt wurde, wuchs die Länge des Programmes und leider auch die Unübersichtlichkeit beträchtlich an, wodurch manche Stellen möglicherweise etwas umständlich programmiert wurden. Doch ich bin sicher, daß dies der Spielfreude keinen Abbruch tun wird. Doch nun viel Spaß bei der Suche nach der goldenen Totenmaske des Pharao.

(Wolfgang Rausch/rg)



Der Programmautor stellt sich vor

Ich bin 28 Jahre alt und gehöre damit laut Statistik der Computerhersteller nicht mehr zu den potentiellen Käufern der Homecomputer. Dies hielt mich jedoch nicht davon ab, Computer zu meinem Hobby zu machen.

Nach einer Lehre als Radio- und Fernsehtechniker war ich acht Jahre bei der Bundeswehr. Zu meiner Entlassung im März 1983 war ein ZX81 das Abschiedsgeschenk meiner Kameraden. Seit April 1983 gehe ich wieder zur Schule und lasse mich zum Computertechniker ausbilden.

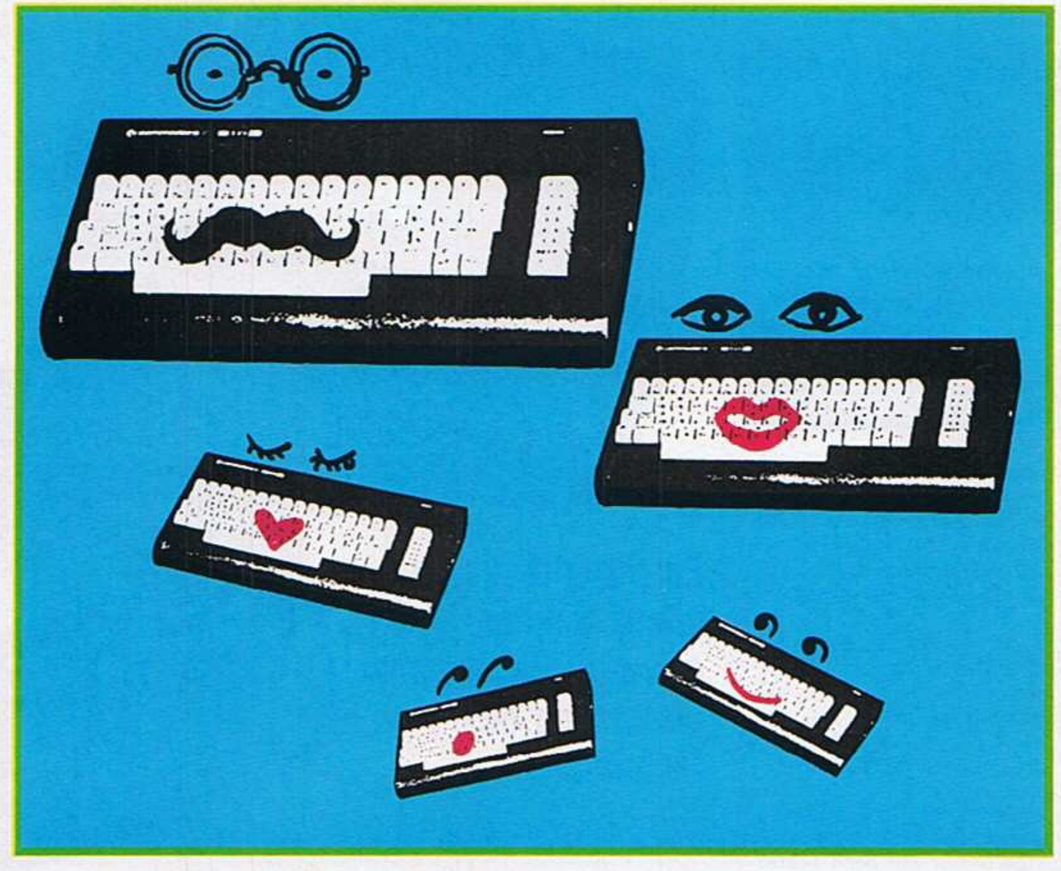
Durch diese Ausbildung stieß ich sehr bald an die Grenzen des ZX81, der ja auch nicht gerade der komfortabelste ist. So legte ich mir nach einem halben Jahr einen VC 20 zu. Zwar auch nicht der Weisheit letzter Schluß, aber dennoch...

Speichererweiterung und Kassetten- Interface baute ich selbst, so daß ein preiswertes, komfortables System entstand. Soweit es die finanziellen Mittel zuließen, folgten inzwischen Floppy, Drukker, Monitor, 40/80 Zeichenkarte und diverse andere Extras. Eine Systemkonfiguration mit der sich schon ganz bequem arbeiten läßt.

(Peter Sprockhoff)

FAMILIENPLANUNG

In der heutigen Zeit ist Empfängnisverhütung und Geburtenkontrolle wohl etwas ganz Alltägliches. Weniger alltäglich ist allerdings der Einsatz eines Homecomputers zu diesem Zweck. Es ist dennoch ein typisches Beispiel für die statistische Auswertung von Meßergebnissen.



Nach dem Motto »Zurück zur Natur« handeln immer mehr Frauen, die auf die herkömmlichen Verhütungsmittel verzichten wollen oder diese aus gesundheitlichen Gründen nicht nehmen. Eine Pillenpause meiner Freundin brachte mich nun auf die Idee, die folgenden Berechnungen in ein Computerprogramm umzusetzen.

Die Methode der natürlichen Verhütung besteht darin, die fruchtbaren Tage eines Zyklus zu errechnen.

Zwei Ärzte, Knaus (Österreich) und Ogino (Japan) fanden voneinander unabhängig eine Verhütungsmethode heraus, die auf dem weiblichen Zyklus basiert.

Zuerst ist es wichtig, den Termin des Eisprunges zu errechnen, da eine Befruchtung nur kurze Zeit nach dem Eisprung möglich ist (etwa 6 Stunden). Knaus und Ogino fanden heraus, daß ziemlich genau 14 Tage vor der nächsten Periode der Eisprung stattfindet. Bezieht man nun die Lebensdauer der Spermen (48 Stunden) in diese Überlegung ein, so kann man die Tage der Enthaltsamkeit eingrenzen. Zählt man zum Termin des Eisprunges 266 Tage (9 Monate) hinzu, so erhält man den Termin einer eventuellen Geburt.

Konkret heißt das: Datum der ersten erfaßten Regel bis zur letzten erfaßten Regel ergibt den Beobachtungszeitraum. Daraus wird der Mittelwert für die Dauer einer Regel gebildet. Zur letzten Regel wird der Mittelwert addiert. Die Standard-Abweichung ergibt die Schwankung um diesen errechneten Termin. Schwankungsmaximum — 14 Tage = letzter Tag der Pause. Schwankungsminimum — 14 Tage — 2 Tage (Lebensdauer Spermen) = erster Tag der Pause. Mittelwert addiert zur letzten Regel — 14 Tage + 266 Tage = Termin einer eventuellen Geburt. Alle Werte werden in Stunden bezogen auf ein festes Datum umgerechnet.

Autor und Redaktion übernehmen keinerlei Haftung bei eventuellen auftretenden Komplikationen.

(Peter Sprockhoff/ev)

Ein heikles Thema

Die einzige Empfängnisverhütung, die die katholische Kirche duldet, heißt Knaus-Ogino oder realistischer Römisches Roulette. Die Methode beruht auf der Messung des Eisprungs. Näheres erfahren Sie vom Hausarzt. Der VC 20 übernimmt die Statistik, aber keine Haftung.

Immer mehr Frauen, die entweder die Pille oder andere Verhütungsmittel nicht vertragen oder nicht nehmen wollen, greifen zu Kalender und Bleistift und errechnen die »gefährlichen« Tage nach der sogenannten Knaus-Ogino-Methode, die auf dem weiblichen Zyklus basiert.

Da es sich um eine statistische Methode handelt, erhöht sich die Sicherheit mit der Anzahl der Messungen. Um das Zahlenmaterial bequem zu verwalten und Rechenfehler mit Folgen weitgehend auszuschalten, bietet sich der Einsatz eines Computers an.

Das Programm ist auf einem VC 20 mit 16-KByte-Erweiterung und Diskettenstation 1541 geschrieben. Es benötigt mit Kommentaren exakt 4614 Byte RAM, kann jedoch durch Fortlassen der REM-Zeilen soweit gekürzt werden, daß es auch auf der Grundversion des VC 20 läuft.

Spezielle POKE-Befehle wurden vermieden, womit das Programm auf allen Commodore-Computern mit minimalen Korrekturen hinsichtlich der Bildschirmaufteilung, lauffähig sein sollte. Die Routinen zum Schreiben und Lesen von sequentiellen Dateien können bei Benutzung einer Datasette sehr leicht geändert werden:

```
1310 OPEN 2,1,0,F$
1380 OPEN 2,1,2,F$
```

Nach dem Starten des Programms meldet es sich zunächst mit einer Kurzanleitung. Anschließend können Daten per Hand eingegeben oder von Diskette gelesen werden. Das Datum muß mit Komma getrennt und die Jahreszahl vierstellig eingegeben werden. Die Uhrzeit ist auf volle Stunden auf- oder abzurunden. Diese Daten werden nämlich für die statistische Aufbereitung der Meßwerte benötigt.

Nach der Eingabe werden die Daten gespeichert. Darauf erfolgt die Ausgabe aller Daten mit der Dauer zwischen der ersten und der n-ten Periode in Tagen sowie der Abweichung in Stunden (Bild 1).

Bei brauchbaren Werten erscheinen jetzt die ersten Ergebnisse, wobei die Werte in Klammern die größte Abweichung nach oben oder unten angeben (Bild 2).

Nach Tastendruck erscheinen schließlich auf dem Bildschirm die wahrscheinlichen Werte der in diesem Zusammenhang interessierenden Ereignisse (Bild 3). Die Empfängniszeit ist in »Pause von ... bis« enthalten. »Nächste P.« ist der voraussichtliche Termin der nächsten Periode. Danach kommen die (voraussichtlichen) Termine des nächsten Eisprunges und einer eventuellen Geburt.

Mit einer einigermaßen zuverlässigen Berechnung ist erst nach mindestens drei eingegebenen Werten zu rechnen. Wie immer bei statistischen Aussagen, so gilt auch hier: Je mehr zuverlässige Werte vorhanden sind, desto genauer wird die Aussage. Autor und Redaktion übernehmen keinerlei Garantie für das einwandfreie Funktionieren dieser Methode.

(Peter Sprockhoff/ev)

Liste der verwendeten Variablen

A$	=	Abweichung in Std.
D$	=	Tag
F$	=	Dateiname
M$	=	Monat
T$	=	Zeit
Y$	=	Jahr
A	=	Feldvariable
B	=	Fehlerquadrate
D	=	Tag
H	=	Stunden
I	=	Laufvariable
J	=	Hilfsvariable
K	=	Laufvariable
M	=	Monat
N	=	Hilfsvariable
S	=	Std. seit 1972
T	=	Uhrzeit
U	=	Abweichung
V	=	Abweichung zu früh
W	=	Abweichung zu spät
X	=	Anzahl der Werte
Y	=	Jahr
Z	=	Arithmetisches Mittel
AA	=	Hilfsvariable

Der Programmaufbau

100	Dimensionierung für 2 Jahre
110- 170	Bedienungsanleitung
180- 200	Eingabe des Namens
220- 280	Überprüfung ob nur 1 Wert vorhanden
290- 370	Eingabe der Daten
380- 450	Maskenaufbau für erste Datenausgabe
460- 540	Berechnung der Periodendauer und Abweichung
550- 600	Ausgabe der Periodendauer und Abweichung
610- 710	Überprüfung ob die Werte geeignet sind
720- 830	Ausgabe der Pause und weiteren Daten
840- 900	Berechnung der Stunden seit 1972
910- 990	Unterroutinen
1000-1070	Berechnung des absoluten Datums und des Jahres
1080-1140	Berechnung des Monats
1150-1240	Berechnung von Tag und Uhrzeit, Umwandlung von Stunden in Tage und Stunden
1250-1280	Berechnung und Ausdruck von Datum und Hinweis
1290-1350	File von Diskette lesen
1360-1420	File auf Diskette schreiben
1430-1610	Umrechnung der Variablen in String

Bild 1. Hardcopy der Werteausgabe

```
BERECHNUNG ( 10 WERTE)

DATUM      |UHR- |DAU |AB-
           |ZEIT |ER  |WEIC
           |     |    |HUNG
----------------------------

1.1.80      12.00
29.1.80     15.00  28   0
26.2.80     18.00  28   0
25.3.80     19.00  28  -2
22.4.80     15.00  27  -3
20.5.80     13.00  27  -5
17.6.80     14.00  28  -2
15.7.80     16.00  28  -1
12.8.80     10.00  27  -9
10.9.80     13.00  29   24

SIND DIE WERTE BRAUCH-
BAR ?

(J/N)? ■
```

Bild 2. Hardcopy der statistischen Kennwerte

```
ERGEBNISSE:

MITTEL 28 TAGE 2 STD.

      (+ 24 /-9 )

STREUUNG 9.1 STD.

BEMERKUNG: WERTE SIND

     SEHR GUT.

 BITTE TASTE DRUECKEN
```

Bild 3. Hardcopy der Ergebnisausgabe

```
PAUSE VON     21.9.80
              4.00 UHR

        BIS   26.9.80
              12.00 UHR

NAECHSTE P.   8.10.80
              15.00 UHR

EISPRUNG      24.9.80
              15.00 UHR

GEBURT        17.6.81
              15.00 UHR


WOLLEN SIE DIE TERMINE
NOCH MAL SEHEN ?

(J/N)? ■
```

Listing »Familienplanung«

```
10 REM**********************                   <053>
20 REM** FAMILIENPLANUNG  **                   <165>
30 REM** FUER VC 20       **                   <130>
40 REM** BEARBEITET VON   **                   <041>
50 REM** PETER SPROCKHOFF **                   <222>
60 REM** ROONSTR. 11      **                   <058>
70 REM** 4790 PADERBORN   **                   <238>
80 REM** 4614 BYTES       **                   <221>
90 REM**********************                   <133>
100 DIM A(24):REM WERTE FUER 2 JAHRE           <000>
110 PRINT"{CLR}** FAMILIENPLANUNG **"          <224>
120 PRINT"{2DOWN}ALLE VERWENDETEN DATENMUESSEN
    ZWISCHEN DEM"                              <169>
130 PRINT"1.1.1973 - 31.12.1999 LIEGEN."       <255>
140 PRINT"{DOWN}DIE REIHENFOLGE DER{3SPACE}
   DATEN DARF NICHT UNTERBROCHEN WERDEN."      <208>
150 PRINT"{DOWN}DIE JAHRESANGABE MUSS 4-STELLIG
    ERFOLGEN."                                 <238>
160 PRINT"{DOWN}SIND DATEN AUF DISK ? "
   :GOSUB 930                                  <209>
170 X=0:FOR I=1 TO 1000:NEXT:PRINT"{CLR}"      <150>
180 IF A=1 THEN INPUT"{DOWN,SPACE}NAME: ";F$
   :GOSUB 1310:GOTO 200                        <197>
190 IF A<>1 THEN INPUT"{DOWN}FUER WEN SOLLEN
    DIE{3SPACE}DATEN ERFASST WERDEN ?{DOWN}";F$
   :GOTO 270                                   <245>
200 PRINT"{DOWN}SOLLEN WEITERE DATEN{2SPACE}
   HINZUGEFUEGT WERDEN ?":GOSUB 930            <000>
210 :                                          <012>
220 REM PRUEFUNG OB NUR 1 WERT                 <208>
230 :                                          <032>
240 IF A<>1 AND X<2 THEN PRINT"{CLR,DOWN}EIN
   WERT IST NICHT{4SPACE}AUSREICHEND !"
   :GOTO 270                                   <145>
250 IF A<>1 THEN 330                           <152>
260 IF X=1 AND A=0 THEN 310                    <216>
270 IF X>=1 AND A=1 THEN PRINT"{CLR}"          <112>
280 :                                          <083>
290 REM EINGABE DATUM & ZEIT MIT "," GETRENNT
                    <059>
300 :                                          <103>
310 X=X+1:PRINT"{DOWN}BITTE DEN"X".TEN WERT
    EINGEBEN."                                 <073>
320 PRINT"TAG,MONAT,JAHR,ZEIT{3SPACE}"
   :INPUT D,M,Y,T:GOSUB 860:A(X)=H:GOTO 200
                    <183>
330 GOSUB 980:IF A<>1 THEN 350                 <083>
340 GOSUB 1380                                 <174>
350 PRINT"{CLR,DOWN}BERECHNUNG ("X"WERTE)":V=0
   :W=0:AA=0                                   <184>
360 Z=(A(X)-A(1))/(X-1):B=0                    <174>
370 :                                          <173>
380 REM MASKENAUFBAU                           <127>
390 :                                          <193>
400 PRINT"{DOWN}DATUM{3SPACE}⊐UHR-⊐DAU⊐AB-"
                    <055>
410 PRINT TAB(8)"⊐ZEIT"TAB(13)"⊐ER"TAB(17)"⊐WEI
   C{8SPACE}⊐{4SPACE}⊐{3SPACE}⊐HUNG"           <195>
420 FOR I=0 TO 21:PRINT TAB(I)"{UP}-":NEXT     <254>
430 FOR K=1 TO X:H=A(K):GOSUB 1030:GOSUB 1470
   :IF K=1 THEN PRINT"{UP}":NEXT K             <152>
440 S=A(K)-A(K-1):PRINT"{UP}
   "TAB(14)INT((S/2.4)/10)                     <209>
450 :                                          <253>
460 REM BERECHNUNG DAUER UND ABWEICHUNG        <114>
470 :                                          <017>
480 U=S-Z:A$=STR$(INT(U)):PRINT"{UP}"TAB(18)A$
                    <067>
490 B=B+U*U                                    <033>
500 IF U>W THEN W=U                            <226>
510 IF U<V THEN V=U                            <236>
520 NEXT K:GOSUB 960:IF A=0 THEN 110           <094>
```

```
530 IF A=1 THEN AA=1                              <207>
540 :                                             <088>
 550 REM AUSGABE DER DAUER UND ABWEICHUNG          <191
560 :                                             <108>
570 PRINT"{CLR,DOWN}ERGEBNISSE:":PRINT
  :W=INT(0.5+W):V=-INT(0.5+ABS(V))                <027>
580 H=Z:GOSUB 1220:PRINT"MITTEL"D;"TAGE";T;
   "STD.";"{DOWN,7SPACE}(+";W;"/";V;")"           <115>
590 A=B/(X-1):A=SQR(A):PRINT"{DOWN}
   STREUUNG"INT(10*A+0.5)/10;"STD."               <012>
600 :                                             <148>
610 REM PRUEFEN OB WERTE GEEIGNET                 <104>
620 :                                             <168>
630 B$="SEHR GUT":IF A>14 THEN B$="GUT"           <005>
640 IF A>24 THEN B$="BEFRIEDIGEND"                <192>
650 IF A>44 THEN B$="MANGELHAFT"                  <075>
660 IF A>54 THEN B$="UNGENUEGEND"                 <180>
670 PRINT"{DOWN}BEMERKUNG: WERTE SIND "           <029>
680 PRINT"{5SPACE}"+B$+".":PRINT                  <134>
690 PRINT"{6DOWN,SPACE}BITTE TASTE DRUECKEN"
                         <065>
700 POKE 198,0:WAIT 198,1                         <030>
710 :                                             <002>
720 REM AUSGABE DER PAUSE UND NAECHSTE DATEN
                         <080>
730 :                                             <022>
740 PRINT"{CLR}":S=A(X)+Z                         <131>
750 H=S-A-410:B$="PAUSE VON{3SPACE}":GOSUB 1270
                         <001>
760 H=S+A-300:B$="{6SPACE}BIS{3SPACE}"
  :GOSUB 1270                                     <117>
770 H=S:B$="NAECHSTE P. ":GOSUB 1270              <066>
780 H=S-336:B$="EISPRUNG{4SPACE}":GOSUB 1270
                         <055>
790 H=S+6048:B$="GEBURT{6SPACE}":GOSUB 1270
                         <210>
800 PRINT:GOSUB 970:IF A=1 THEN 350               <043>
810 IF A=0 THEN CLR:PRINT"{CLR}"SPC(225)"DANKE
   SCHOEN":END                                    <081>
820 END                                           <183>
830 :                                             <123>
840 REM BERECHNUNG DER STUNDEN SEIT 1.1.1972 0.
   00 UHR                                         <042>
850 :                                             <143>
860 Y=Y-1972:IF Y<1 OR Y>27 THEN PRINT"JAHR
   IST UNZULAESSIG !":STOP                        <068>
870 H=24*(365*Y+INT(Y/4)+D)+T
  :IF(INT(Y/4)=Y/4)AND M<3 THEN H=H-24            <093>
880 FOR I=1 TO M:READ D:NEXT:RESTORE:H=H+24*D
  :RETURN                                         <220>
890 DATA 0,31,59,90,120,151,181,212,243,273,304,
  334,365                                         <216>
900 :                                             <193>
910 REM FRAGE MIT JA ODER NEIN BEANTWORTEN        <151>
920 :                                             <213>
930 J=1:N=0:INPUT"{DOWN}(J/N)";A$
  :IF A$="J"THEN A=1:RETURN                       <005>
940 IF A$="N"THEN A=0:RETURN                      <117>
950 PRINT"{DOWN}MIT J ODER N ANTWORTEN":GOTO 930
                         <116>
960 PRINT"{DOWN}SIND DIE WERTE BRAUCH-BAR ?"
  :GOSUB 930:RETURN                               <089>
970 PRINT"{DOWN}WOLLEN SIE DIE TERMINENOCH MAL
   SEHEN ?":GOSUB 930:RETURN                      <050>
980 PRINT"{DOWN}SOLLEN DATEN ABGESPEI-CHERT
   WERDEN ?":GOSUB 930:RETURN                     <016>
990 :                                             <027>
1000 REM BERECHNUNG DES ABSOLUTEN DATUMS AUS H
   (SEIT 1972)                                    <060>
1010 REM BERECHNUNG DES JAHRES                    <254>
1020 :                                            <057>
1030 Y=1972:H=H-8784:IF H<0 THEN PRINT"FEHLER
   !":STOP                                        <255>
1040 Y=Y+1:IF H<8760 THEN 1100                    <081>
1050 IF INT(Y/4)<>Y/4 THEN H=H-8760:GOTO 1040
                         <120>
1060 IF H>=8784 THEN H=H-8784:GOTO 1040           <048>
1070 :                                            <108>
1080 REM BERECHNUNG DES MONATS                    <090>
1090 :                                            <128>
1100 M=-1                                         <043>
1110 M=M+1:N=J:READ J:IF M>1 AND INT(Y/4)=Y/4 T
   HEN J=J+1                                      <167>
1120 IF H>=J*24 THEN 1110                         <096>
1130 RESTORE:H=H-N*24                             <129>
1140 :                                            <178>
1150 REM BERECHNUNG VON TAG UND UHRZEIT           <211>
1160 :                                            <198>
1170 GOSUB 1220:D=D+1                             <055>
1180 RETURN                                       <046>
1190 :                                            <228>
1200 REM UMWANDLUNG VON STUNDEN IN TAGE UND STU
   NDEN                                           <025>
1210 :                                            <248>
1220 IF H>2000 OR H<0 THEN PRINT"{DOWN}STUNDEN
    AUSSERHALB DESBEREICHS !":STOP                <149>
1230 D=INT(H/24):T=INT(H-24*D):RETURN             <128>
1240 :                                            <022>
1250 REM BERECHNUNG UND AUSDRUCK VON DATUM MIT
   HINWEIS                                        <014>
1260 :                                            <042>
1270 GOSUB 1030:GOSUB 1460:PRINT:RETURN           <120>
1280 :                                            <063>
1290 REM FILE VON DISKETTE LESEN                  <133>
1300 :                                            <083>
1310 OPEN 2,8,2,F$+",S,R"                         <055>
1320 INPUT#2,F$:INPUT#2,X                         <237>
1330 FOR J=1 TO X:INPUT#2,A(J):NEXT               <149>
1340 CLOSE 2:RETURN                               <219>
1350 :                                            <133>
1360 REM FILE AUF DISKETTE SCHREIBEN              <208>
1370 :                                            <153>
1380 OPEN 2,8,2,"@:"+F$+",S,W"                    <234>
1390 PRINT#2,F$:PRINT#2,X                         <091>
1400 FOR J=1 TO X:PRINT#2,A(J):NEXT               <239>
1410 CLOSE 2:RETURN                               <033>
1420 :                                            <203>
1430 REM AUSDRUCK VON DATUM UND HINWEIS           <248>
1440 REM UMRECHNUNG VON D,M,Y,T IN STRING         <083>
1450 :                                            <233>
1460 PRINT B$;                                    <243>
1470 D$=STR$(D):M$=STR$(M):Y$=STR$(Y-1900)
  :T$=STR$(INT(T+0.5)):K$=".":U$=".00"            <085>
1480 Y$=RIGHT$(Y$,2)                              <241>
1490 IF LEN(T$)=2 THEN T$=RIGHT$(T$,1)            <146>
1500 IF LEN(T$)>2 THEN T$=RIGHT$(T$,2)            <156>
1510 IF LEN(D$)=3 AND LEN(M$)=2 THEN 1550         <097>
1520 IF LEN(D$)=2 AND LEN(M$)=3 THEN 1560         <108>
1530 IF LEN(D$)=3 AND LEN(M$)=3 THEN 1570         <120>
1540 D$=RIGHT$(D$,1):M$=RIGHT$(M$,1):GOTO 1580
                         <217>
1550 D$=RIGHT$(D$,2):M$=RIGHT$(M$,1):GOTO 1580
                         <228>
1560 D$=RIGHT$(D$,1):M$=RIGHT$(M$,2):GOTO 1580
                         <238>
1570 D$=RIGHT$(D$,2):M$=RIGHT$(M$,2):GOTO 1580
                         <249>
1580 IF AA=1 THEN 1600                            <144>
1590 PRINT D$+K$+M$+K$+Y$TAB(9)T$+U$:RETURN
                         <025>
1600 PRINT D$+K$+M$+K$+Y$                         <187>
1610 PRINT TAB(12)T$+U$;" UHR":RETURN             <233>
```

Listing »Familienplanung« (Schluß)

Das Grab des Pharao

Fortsetzung von Seite 51

Auf der Suche nach der goldenen Totenmaske des Pharao müssen Sie hier die erste Aufgabe lösen. Eine Menge Programmzeilen sind auf dem Weg in die Wüste, in der das Grab des Pharao steht, einzutippen. Damit Sie nicht hier schon »in der Wüste der Fehlermeldungen« stehen, ist auch dieses Programm mit dem Checksummer einzugeben. Liegt diese Hürde hinter Ihnen, wird Ihnen ein freundlicher Beduine, sobald Sie Ihn gefunden haben, den richtigen Weg weisen. (rg)

Variablenliste »Grab des Pharao«

PN (58,6)	= Spielfeld (besteht aus 58 Bildern)
IT$ (54)	= Liste der Gegenstände (Namen) und Wörter (zur Verschlüsselung der Kommentare)
IN (46,3)	= Information über Standort der Gegenstände und Möglichkeit diese mitzunehmen
CO$ (18)	= Liste der Verben
AP	= momentaner Standpunkt
X1,X2,X4	= Flags für bestimmte Spielsituationen
X5,X6,X9	= Flags für bestimmte Spielsituationen
Y1,Y2,Y3,Y4	= Flags für bestimmte Spielsituationen
GE	= Anzahl der Gegenstände, die man bei sich trägt (maximal 3)

Programmbeschreibung »Grab des Pharao«

10—13	: Maschinenprogramm zur Verschiebung des Zeichengenerators
30—150	: Einlesen der Felder (Arrays)
164—410	: Information zu den Bildern und Eingabe eines Verbs
500—1910	: Befehlsauswertung
5000—5910	: Kommentare und Eingabe eines Nomens
10000—10900	: Einlesen der Charakter-DATA
12000—12167	: Einlesen der Sprite-DATA
19000—20900	: Zeichnen der Bilder (Hintergrund)
21000—21018	: Zeichnen der Durchgänge
21030—21038	: Zeichnen der Türen (Sprites)
21050—21500	: Zeichnen der Gegenstände (hauptsächlich Sprites)
30000—30040	: Gegenstände, die man trägt, auflisten
34000—35060	: Routinen zum Einlesen und Abspeichern eines Files
39900—40044	: DATAs für Verben, Nomen, Spielfeld und Gegenstände
45000—49065	: Vorspann
50105—52030	: Kommentare bei Niederlage des Spielers und Routinen für bewegliche Gegenstände

Listing »Grab des Pharao«. Beachten Sie beim Eintippen bitte den Beitrag über den »Checksummer 64«

```
0 REM *** PHARAOS GRAVE V.2 ***                 <196>
1 REM      BY W.RAUSCH                          <118>
2 REM      MOERIKEWEG 73                        <234>
3 REM      8504 STEIN                           <230>
5 A1$="{HOME,18DOWN}"                           <214>
6 A2$="{HOME,21DOWN}"                           <011>
7 A3$="{HOME,22DOWN}"                           <030>
8 A4$="{HOME,23DOWN}"                           <049>
9 PRINT"{CLR}":V=53248:SI=54272:POKE V+21,0
   :POKE 657,128                                <023>
10 FOR I=832 TO 865:READ A:POKE I,A:NEXT        <175>
11 DATA 120,169,51,133,1,169,0,133,95           <036>
12 DATA 133,90,133,88,169,208,133,96,169,240,
   133,89,169,224,133,91,32,191,163             <255>
13 DATA 169,55,133,1,88,96                      <029>
15 SYS 832:POKE 850,160:SYS 832
   :POKE 56576,PEEK(56576)AND 252:POKE 53272,8
                    <154>
16 POKE 648,192                                 <017>
20 GOTO 10000                                   <142>
30 DIM PN(58,6),IT$(54),IN(46,3),CO$(18)        <130>
40 FOR I=1 TO 18:READ CO$(I):NEXT               <175>
50 FOR I=1 TO 54:READ IT$(I):NEXT               <196>
55 RETURN                                       <197>
60 FOR I=866 TO 895:READ A:POKE I,A:NEXT
   :IF KK=1 THEN RETURN                         <094>
65 FOR I=1 TO 58                                <255>
70 READ PN(I,1),PN(I,2),PN(I,3),PN(I,4),PN(I,5),
   PN(I,6)                                      <054>
80 NEXT                                         <210>
90 FOR I=1 TO 46                                <021>
100 READ IN(I,1),IN(I,2),IN(I,3)                <240>
110 NEXT                                        <240>
130 RETURN                                      <016>
150 AP=1                                        <010>
155 POKE 53280,11:POKE 53281,0                  <242>
160 PRINT"{CLR}":POKE V+21,0
   :POKE V+17,PEEK(V+17)AND 239:GOSUB 19000
                    <142>
164 POKE 214,15:SYS 58640:PRINT"{RED}
   WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
   {WHITE}"                                     <115>
165 POKE V+17,PEEK(V+17)OR 16                   <098>
170 PRINT"{UP}MOEGLICHE RICHTUNGEN :";          <017>
174 FOR I=1 TO 6                                <053>
175 IF PN(AP,I)>0 THEN ON I GOSUB 179,180,181,
   182,184,185                                  <164>
176 NEXT:GOTO 210                               <136>
179 PRINT" W";:RETURN                           <234>
180 PRINT" O";:RETURN                           <227>
181 PRINT" N";:RETURN                           <227>
182 PRINT" S";:RETURN                           <233>
184 PRINT" H";:RETURN                           <224>
185 PRINT" R":RETURN                            <176>
210 PRINT A1$"BESONDERHEITEN: ";:N=0            <051>
220 FOR I=1 TO 46                               <151>
222 IF N>0 AND IN(I,1)=AP AND IN(I,
   3)=1 THEN PRINT"{16RIGHT}"IT$(I)             <233>
230 IF IN(I,1)=AP AND N=0 AND IN(I,
   3)=1 THEN PRINT IT$(I):N=1                   <073>
240 NEXT                                        <114>
242 IF AP=35 AND X8=1 THEN 50100                <231>
245 IF AP=16 AND IN(14,1)=-1 AND IN(15,
   1)>0 THEN 50800                              <240>
246 IF AP=53 AND IN(45,1)=-1 THEN 50720         <188>
247 IF AP=50 AND IN(46,1)<>-1 THEN 50207        <109>
250 PRINT A3$:INPUT"KOMMANDO ";B$               <037>
260 FOR I=1 TO 18                               <191>
270 IF LEFT$(B$,4)=LEFT$(CO$(I),4)THEN 300      <046>
280 NEXT                                        <155>
300 IF AP=41 AND IN(41,1)<>-1 OR AP=42 AND IN(4
   1,1)<>-1 THEN GOSUB 52000                    <022>
310 IF C=3 THEN 50208                           <144>
410 IF AP=24 THEN GOTO 50204                    <250>
500 IF I<>1 THEN 730                            <159>
510 GOSUB 5540                                  <090>
530 GOTO 550                                    <055>
540 GOTO 5530                                   <116>
550 IF AP=2 OR AP=3 THEN X1=X1+1                <148>
560 IF X1=2 THEN GOTO 50190                     <089>
570 IF AP=10 AND IN(6,1)=-1 THEN 5530           <151>
590 IF AP=19 AND A$="R"AND IN(9,
   1)<>19 THEN GOTO 50203                       <009>
610 IF AP=31 AND A$="S"THEN GOTO 50202          <028>
```

```
620 IF AP=48 AND A$="S"OR AP=44 AND A$="W"THEN
   50206                                        <101>
630 IF AP=35 AND IN(14,1)<>AP THEN 50100        <153>
640 IF AP=37 AND A$="O"AND IN(30,
   1)=37 THEN AP=41:GOTO 5500                   <098>
645 IF AP=37 AND IN(30,1)<>37 AND A$="O"THEN 50
   200                                          <219>
650 IF AP=50 AND A$="W"AND IN(31,1)<>0 THEN 5530
                    <128>
655 IF AP=38 AND A$="N"AND IN(28,1)=0 AND IN(46,
   1)<>-1 THEN 50207                            <165>
660 IF A$="W"AND PN(AP,1)<>0 THEN AP=PN(AP,1)
   :GOTO 5500                                   <071>
670 IF A$="O"AND PN(AP,2)<>0 THEN AP=PN(AP,2)
   :GOTO 5500                                   <075>
680 IF A$="N"AND PN(AP,3)<>0 THEN AP=PN(AP,3)
   :GOTO 5500                                   <086>
690 IF A$="S"AND PN(AP,4)<>0 THEN AP=PN(AP,4)
   :GOTO 5500                                   <103>
700 IF A$="H"AND PN(AP,5)<>0 THEN AP=PN(AP,5)
   :GOTO 5500                                   <104>
710 IF A$="R"AND PN(AP,6)<>0 THEN AP=PN(AP,6)
   :GOTO 5500                                   <126>
720 GOTO 5530                                   <040>
730 IF I<>2 THEN 830                            <136>
735 IF IN(1,1)<>-1 THEN 5530                    <150>
739 X2=X2+1:IF X2>5 THEN 5475                   <055>
740 GOSUB 5540                                  <065>
770 IF AP=7 AND IN(3,1)=0 AND IN(4,
   1)=AP AND A$="EINGA"THEN IN(4,1)=0:GOTO 5410
                    <123>
780 IF AP=13 AND A$="TUER"THEN GOTO 50000       <176>
790 IF AP=13 AND A$="WAND"AND IN(11,
   1)=0 THEN IN(11,1)=13:GOTO 5420              <169>
800 IF AP=21 AND IN(16,1)>0 AND A$=IT$(16)THEN
   IN(16,1)=0:GOTO 5500                         <113>
810 IF AP=54 AND IN(40,1)>0 AND A$="AUSGA"THEN
   GOTO 49000                                   <069>
820 GOTO 5530                                   <141>
830 IF I<>3 THEN 940                            <240>
840 GOSUB 5540:GOSUB 5000                       <050>
850 IF X<>32 AND X<>40 AND X<>38 THEN GOTO 5530
                    <163>
860 IF X<>32 THEN 920                           <077>
870 IF AP=15 THEN PN(15,3)=16:IN(13,1)=0
   :GOTO 5500                                   <099>
875 IF AP=13 THEN 50211                         <064>
880 IF AP=21 AND IN(16,1)=0 THEN PN(21,3)=22
   :IN(17,1)=0:GOTO 5500                        <165>
881 IF AP=25 AND IN(43,1)=25 THEN IN(43,1)=0
   :PN(25,4)=19:IN(42,1)=0:PN(19,3)=25:GOTO 5500
                    <189>
882 IF AP=31 AND X5=1 AND IN(23,
   1)=AP THEN IN(23,1)=0:PN(31,3)=32:GOTO 5500
                    <037>
890 IF AP=21 AND IN(16,1)>0 THEN 50213          <140>
895 IF X5=1 AND AP=31 THEN PN(AP,3)=32
   :IN(31,1)=0:GOTO 5500                        <194>
900 IF AP=45 THEN PN(45,3)=50:IN(28,1)=0
   :GOTO 5500                                   <139>
905 IF AP=42 THEN PN(42,3)=40:IN(26,1)=0
   :GOTO 5500                                   <135>
906 IF AP=47 AND PN(AP,3)=0 THEN PN(AP,3)=52
   :IN(36,1)=0:GOTO 5500                        <044>
910 IF AP=38 THEN PN(38,3)=50:IN(28,1)=0
   :GOTO 5500                                   <153>
920 IF AP=50 THEN PN(50,1)=44:IN(31,1)=0
   :GOTO 5500                                   <146>
925 IF AP=53 AND Y3=0 AND X=38 THEN Y3=1
   :GOTO 50700                                  <236>
926 IF AP=57 AND X=40 AND Y4=0 AND IN(44,
   1)=-1 THEN IN(41,1)=AP:Y4=1:GOTO 5458        <215>
930 GOTO 5530                                   <251>
940 IF I<>4 THEN 1130                           <135>
950 GOSUB 5540                                  <020>
955 IF AP<8 OR A$<>IT$(53)THEN GOTO 5530        <181>
960 PRINT A2$:INPUT"{GREEN}WELCHE
   : W O N S{3SPACE}";C$                        <239>
970 IF AP=13 AND IN(11,1)=0 AND C$="O"THEN GOTO
   5430                                         <142>
980 IF AP<>13 OR AP=13 AND C$<>"O"THEN 1100
                    <136>
1000 GOTO 5530                                  <065>
1100 PRINT A3$"{GREEN}NICHTS PASSIERT{7SPACE,
   WHITE}":GOSUB 5800:GOSUB 5900:GOTO 164       <163>
1130 IF I<>5 THEN 1180                          <076>
1140 GOSUB 5540                                 <211>
1150 GOSUB 5000                                 <212>
1160 IF AP=5 AND X=2 AND IN(X,1)>0 THEN IN(2,
   1)=0:IN(X,2)=0:GOTO 5400                     <237>
1170 GOTO 5530                                  <236>
1180 IF I<>6 THEN 1220                          <122>
1182 IF IN(1,1)<>-1 THEN 5530                   <087>
1183 X2=X2+1:IF X2>5 THEN 5475                  <245>
1195 GOSUB 5540:GOSUB 5000                      <140>
1200 IF AP=7 AND IN(X,1)=AP AND IN(X,
   1)>0 THEN IN(X,1)=0:IN(4,3)=1:GOTO 5500      <192>
1210 GOTO 5530                                  <020>
1220 IF I<>7 THEN 1270                          <168>
1230 GOSUB 5540:GOSUB 5000                      <185>
1240 IF X<>15 AND X<>5 THEN 5530                <157>
1250 IF AP=16 AND IN(X,1)=AP THEN IN(X,1)=0
   :GOTO 5500                                   <093>
1255 IF AP=35 AND X=5 THEN 50100                <172>
1260 GOTO 5530                                  <070>
1270 IF I<>8 THEN 1320                          <215>
1280 GOSUB 5540                                 <096>
1285 IF AP=7 AND IN(4,1)=0 AND A$=LEFT$(IT$(11),
   5)THEN AP=8:GOTO 5500                        <124>
1290 IF AP=13 AND A$=LEFT$(IT$(51),5)AND IN(11,
   1)=AP THEN AP=14:GOTO 5500                   <110>
1300 IF AP=14 AND A$=LEFT$(IT$(51),5)THEN AP=13
   :GOTO 5500                                   <223>
1310 GOTO 5530                                  <121>
1320 IF I<>9 THEN 1360                          <015>
1330 GOSUB 5540:GOSUB 5000                      <030>
1335 IF X<>9 AND IN(X,1)<>-1 THEN 5530          <188>
1340 IF AP=51 AND PN(AP,5)=0 THEN PN(AP,5)=54
   :IN(X,1)=AP:GOTO 5440                        <052>
1345 IF AP=37 THEN IN(X,1)=0:GOTO 5470          <055>
1350 GOTO 5530                                  <161>
1360 IF I<>10 THEN 1410                         <091>
1370 GOSUB 5540                                 <186>
1375 IF AP=27 THEN 1395                         <020>
1376 GOSUB 5000                                 <183>
1390 IF AP=40 AND Y2=0 AND IN(X,1)=AP THEN Y2=1
   :IN(35,1)=AP:PN(40,3)=47:GOTO 5450           <041>
1395 IF AP=27 AND A$="GRABR"AND IN(46,
   3)=0 THEN IN(46,3)=1:GOTO 5480               <056>
1400 GOTO 5530                                  <211>
1410 IF I<>11 THEN 1460                         <147>
1420 GOSUB 5540:GOSUB 5000                      <120>
1430 IF AP=10 AND X=7 AND IN(6,1)=-1 THEN 50206
                    <022>
1440 IF AP=52 AND X=37 AND IN(34,
   1)=AP THEN PN(52,3)=53:IN(34,1)=0:GOTO 1680
                    <107>
1450 GOTO 5530                                  <005>
1460 IF I<>12 THEN 1510                         <194>
1470 GOSUB 5540                                 <030>
1490 IF X4=1 THEN AP=54:GOTO 5500               <177>
1500 GOTO 5530                                  <055>
1510 IF I<>13 THEN 1580                         <252>
1520 GOSUB 5540                                 <080>
1525 IF GE>=3 THEN 5460                         <029>
1530 GOSUB 5000                                 <081>
1535 IF AP=16 AND X=14 THEN IN(X,1)=-1
   :IN(15,3)=1:GOTO 5500                        <128>
1536 IF AP=10 AND X=6 THEN IN(7,3)=1            <044>
1540 IF IN(X,1)=AP AND IN(X,2)>0 THEN IN(X,1)=-1
   :GOTO 5500                                   <002>
1550 IF IN(X,2)=0 THEN 5530                     <147>
1560 IF IN(X,1)<>AP THEN PRINT A3$"{GREEN}NICHT
   VORHANDEN{WHITE}":GOSUB 5800:GOSUB 5900
   :GOTO 164                                    <028>
1570 GOTO 5530                                  <126>
1580 IF I<>14 THEN 1640                         <065>
1590 GOSUB 5540                                 <151>
1600 GOSUB 5000                                 <152>
1610 IF IN(X,1)<>-1 THEN PRINT A3$"{GREEN}ICH
   HABE DAS NICHT{WHITE}":GOSUB 5800:GOSUB 5900
   :GOTO 164                                    <176>
1612 IF AP=19 AND A$="SEIL"AND IN(9,
   1)=-1 THEN GOSUB 5455:GOTO 1620              <015>
1615 IF AP=31 AND A$="DIAMA"AND IN(X,
   1)=-1 THEN X5=1:GOSUB 5465:IN(20,2)=0
```

Listing »Grab des Pharao« (Fortsetzung)

```
   :IN(20,3)=0:GOTO 1620                          <142>
1620 IN(X,1)=AP:GOTO 5500                         <199>
1640 IF I<>15 THEN 1700                           <123>
1650 GOSUB 5540:GOSUB 5000                        <095>
1655 IF X<>7 THEN 5530                            <111>
1660 IF AP=10 AND IN(6,1)=-1 AND X9=0 THEN PN(A
   P,1)=11:IN(8,1)=0:X9=1:GOTO 1680               <252>
1670 GOTO 5530                                    <226>
1680 PRINT A3$"{GREEN}DIE "IT$(32)" OEFFNET
    SICH{WHITE}":GOSUB 5800:GOTO 160              <215>
1700 IF I<>16 THEN 1750                           <189>
1710 GOTO 30000                                   <048>
1750 IF I<>17 THEN 1800                           <236>
1755 GOTO 34000                                   <097>
1800 IF I<>18 THEN 1900                           <033>
1810 POKE V+21,0:PRINT CHR$(14),"{CLR,2DOWN,
   3RIGHT}SIE WOLLEN AUFGEBEN (J/N) ?"            <001>
1820 GET A$:IF A$="J"THEN SYS 64738               <228>
1830 IF A$="N"THEN PRINT CHR$(142):GOSUB 5600
   :GOTO 160                                      <060>
1840 IF A$=""OR A$=" "THEN 1820                   <154>
1900 PRINT A4$"UNBEKANNTES WORT":GOSUB 5800
                         <007>
1910 PRINT A3$"{17SPACE}":PRINT A4$"{24SPACE}"
   :GOTO 164                                      <000>
5000 FOR X=1 TO 46                                <101>
5010 IF LEFT$(A$,5)=LEFT$(IT$(X),5)THEN RETURN
                         <218>
5020 NEXT                                         <049>
5030 PRINT A4$;TAB(20)"{CYAN}UNBEKANNTES WORT
   {WHITE}":FOR T=1 TO 1000:NEXT                  <217>
5035 GOSUB 5900                                   <025>
5040 X=0:GOTO 164                                 <091>
5400 PRINT A3$"{GREEN}GEHE IN RICHTUNG DES "IT$
   (50)"{WHITE}":GOSUB 5800                       <004>
5401 PN(5,2)=6:GOTO 160                           <242>
5410 PRINT A3$"{GREEN}DURCHGANG IN DER MAUER
   {WHITE}":IN(4,1)=0:X6=1:GOSUB 5800:GOTO 5500
                         <175>
5420 PRINT A3$"{GREEN}EIN "IT$(51)" WIRD SICHTB
   AR{WHITE}":GOSUB 5800:GOTO 160                 <229>
5430 PRINT A3$"{GREEN}ES KLINGT HOHL{11SPACE,
   WHITE}":GOSUB 5800:GOTO 160                    <093>
5440 PRINT A3$"{GREEN}"IT$(9)" HAT SICH OBEN
    VERFANGEN{WHITE}":X4=1:GOSUB 5800:GOTO 160
                         <215>
5450 PRINT A3$"{GREEN}EIN "IT$(11)" WIRD SICHTB
   AR{WHITE}":GOSUB 5800:GOTO 160                 <255>
5455 PRINT A3$"{GREEN}"IT$(9)" HAENGT HINUNTER
   {WHITE}":GOSUB 5800:RETURN                     <011>
5458 PRINT A3$"{GREEN}ER ENTHAELT EIM "IT$(41)"
   {WHITE,INST}":GOSUB 5800:GOTO 160              <003>
5460 PRINT A3$"{GREEN}SIE KOENNEN NICHT SO VIEL
    TRAGEN{WHITE}":GOSUB 5800:GOTO 160            <231>
5465 PRINT A3$"{GREEN}"IT$(20)" IST DAS FEHLEND
   E "IT$(52)" DER "IT$(6)"{WHITE}":GOSUB 5800
   :RETURN                                        <065>
5470 PRINT A3$"{GREEN}"IT$(9)" IST HINUNTERGEFA
   LLEN{WHITE}":GOSUB 5800:GOTO 160               <035>
5475 PRINT A3$"{GREEN}"RIGHT$(IT$(1),5)" IST
    JETZT STUMPF{WHITE}":GOSUB 5800:GOSUB 5900
   :GOTO 164                                      <249>
5480 PRINT A3$"{GREEN}ER TRUG EINE "IT$(46)"
    BEI SICH{WHITE}":GOSUB 5800:GOTO 160          <180>
5500 GOSUB 5900                                   <236>
5510 PRINT A4$"{CYAN}IN ORDNUNG{WHITE}"
   :FOR T=1 TO 1000:NEXT:IF I=13 THEN GE=GE+1
                         <236>
5520 IF I=14 THEN GE=GE-1                         <221>
5525 GOSUB 5600:GOTO 160                          <092>
5530 GOSUB 5900                                   <010>
5535 PRINT A4$"{CYAN}NICHT MOEGLICH{WHITE}"
   :FOR T=1 TO 1000:NEXT:PRINT A4$"{15SPACE}"
   :GOTO 164                                      <173>
5540 PRINT A4$;TAB(25):INPUT A$:A$=LEFT$(A$,5)
   :RETURN                                        <117>
5600 POKE V+21,0:POKE V+28,0:POKE V+23,0
   :POKE V+29,0:RETURN                            <212>
5800 FOR T=1 TO 3000:NEXT:RETURN                  <097>
5900 PRINT A3$"{39SPACE}"                         <152>
5910 PRINT A4$"{38SPACE}":RETURN                  <107>
10000 AD=57344                                    <117>
10010 READ X:IF X=-1 THEN GOTO 45000              <244>
10020 CG=AD+X*8:FOR I=0 TO 7:READ Q:POKE CG+I,Q
   :NEXT:GOTO 10010                               <001>
10110 DATA 28,252,243,207,63,255,255,255,255
                         <150>
10120 DATA 30,255,255,255,255,252,243,207,63
                         <153>
10130 DATA 31,255,255,255,255,255,255,15,240
                         <164>
10140 DATA 33,255,255,255,255,15,240,255,255
                         <176>
10150 DATA 35,255,255,15,240,254,254,254,254
                         <184>
10160 DATA 36,0,31,227,252,254,254,254,254 <087>
10170 DATA 37,255,255,255,255,63,207,243,252
                         <210>
10180 DATA 38,63,207,243,252,255,255,255,255
                         <221>
10190 DATA 39,0,248,199,63,127,127,127,127 <134>
10200 DATA 40,255,255,240,15,255,255,255,255
                         <234>
10210 DATA 41,255,255,255,255,240,14,254,254
                         <242>
10220 DATA 42,255,255,255,255,255,255,240,15
                         <000>
10230 DATA 43,254,253,251,247,239,223,191,127
                         <059>
10240 DATA 44,254,254,254,254,254,254,254,254
                         <075>
10250 DATA 45,127,127,127,127,127,127,127,127
                         <078>
10260 DATA 46,255,255,255,255,255,255,255,255
                         <105>
10270 DATA 47,0,255,255,255,255,255,255,255
                         <008>
10280 DATA 48,0,254,254,254,254,254,254,254
                         <012>
10290 DATA 49,0,127,127,127,127,127,127,127
                         <016>
10300 DATA 50,127,191,223,239,247,251,253,254
                         <128>
10400 DATA 51,239,239,239,239,239,223,191,127
                         <243>
10410 DATA 52,224,239,239,239,239,239,239,239
                         <006>
10420 DATA 53,239,239,239,143,47,239,239,239
                         <222>
10430 DATA 54,0,239,239,239,239,239,239,239
                         <180>
10440 DATA 55,254,254,254,254,254,224,235,239
                         <020>
10450 DATA 56,255,255,255,255,255,255,95,241
                         <245>
10460 DATA 57,254,254,254,254,0,254,254,254
                         <192>
10470 DATA 27,239,239,239,239,239,239,239,224
                         <068>
10480 DATA 59,239,239,239,239,247,251,253,254
                         <075>
10490 DATA 60,239,239,239,239,239,239,239,15
                         <035>
10500 DATA 61,239,239,227,233,236,239,239,239
                         <091>
10510 DATA 62,239,239,239,239,15,239,239,239
                         <058>
10520 DATA 29,15,239,239,239,239,239,239,239
                         <071>
10530 DATA 64,127,127,127,127,0,127,127,127
                         <254>
10540 DATA 65,255,255,255,255,255,15,239,239
                         <081>
10550 DATA 67,0,255,255,0,0,0,0,0          <007>
10560 DATA 68,239,239,239,239,239,239,239,239
                         <170>
10570 DATA 69,254,254,254,254,254,254,240,15
                         <099>
10580 DATA 70,239,239,239,239,240,239,239,239
                         <175>
10590 DATA 71,254,254,254,254,254,254,254,0
                         <063>
10600 DATA 72,127,127,127,127,127,127,127,0
                         <067>
10610 DATA 73,15,240,255,255,255,255,255,255
                         <140>
```

Listing »Grab des Pharao« (Fortsetzung)

```
10620 DATA 74,240,15,255,255,255,255,255,255
                          <151>
10630 DATA 75,255,170,255,170,255,170,255,170
                          <206>
10640 DATA 76,85,85,85,85,85,87,95,255      <163>
10650 DATA 77,85,85,85,87,127,255,255,255   <056>
10660 DATA 78,85,85,85,85,245,255,255,255   <066>
10670 DATA 79,85,85,85,85,85,245,255,255    <030>
10680 DATA 80,85,85,85,85,85,85,245,255     <241>
10685 DATA 81,63,207,243,252,254,254,254,254
                          <210>
10690 DATA 82,254,254,254,254,15,240,255,255
                          <216>
10695 DATA 83,127,255,15,240,255,255,255,255
                          <224>
10700 DATA 84,252,242,206,62,254,254,254,254
                          <225>
10705 DATA 85,254,254,254,254,252,243,207,63
                          <234>
10710 DATA 86,255,255,255,255,252,242,206,62
                          <241>
10715 DATA 87,255,255,255,255,240,14,254,254
                          <247>
10720 DATA 88,254,254,254,254,254,254,240,15
                          <250>
10725 DATA 89,254,254,240,15,255,255,255,255
                          <004>
10730 DATA 90,126,190,222,238,246,250,252,254
                          <044>
10735 DATA 91,127,127,127,127,63,207,243,252
                          <001>
10740 DATA 92,14,240,255,255,255,255,255,255
                          <014>
10745 DATA 93,96,99,105,108,110,110,110,110
                          <201>
10750 DATA 94,6,198,150,54,118,118,118,118 <182>
10755 DATA 95,110,110,110,110,110,110,110,110
                          <014>
10760 DATA 97,118,118,118,118,118,118,118,118
                          <085>
10765 DATA 98,110,110,110,110,108,105,99,15
                          <218>
10770 DATA 99,118,118,118,118,54,150,198,240
                          <048>
10775 DATA 100,0,0,0,0,0,15,63,127          <014>
10780 DATA 101,15,31,63,127,255,255,255,255
                          <248>
10785 DATA 102,240,248,252,254,255,255,255,255
                          <153>
10790 DATA 103,0,0,0,0,240,252,254          <129>
10795 DATA 104,168,168,168,168,168,168,0,0  <231>
10800 DATA 105,170,170,170,170,170,170,0,0  <195>
10805 DATA 106,168,168,168,168,168,168,168,168
                          <209>
10810 DATA 107,162,162,162,162,162,162,162,162
                          <167>
10815 DATA 108,0,0,168,168,168,168,168 <255>
10820 DATA 109,170,170,170,170,170,170,170,170
                          <171>
10825 DATA 110,0,0,170,170,170,170,170,170 <216>
10830 DATA 111,0,0,129,195,255,255,255,255 <249>
10835 DATA 112,255,255,248,6,184,191,255,255
                          <112>
10840 DATA 113,0,0,1,3,15,255,255,255       <248>
10845 DATA 114,0,0,128,192,224,255,255,255 <003>
10850 DATA 115,0,0,0,255,255,0,0,0          <094>
10855 DATA 116,193,182,182,128,193,227,193,193
                          <236>
10860 DATA 117,129,129,129,129,255,255,255,255
                          <242>
10865 DATA 118,255,219,129,195,195,129,219,255
                          <254>
10900 DATA-1                                <159>
12000 FOR I=0 TO 62:READ Q:POKE 50816+I,Q:NEXT
                          <217>
12001 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         <250>
12002 DATA 0,0,64,0,0,128,0,1,0,0,3,255,248,4,0,
   0,8,0,0,8,0,0,8,0,0,0,0,0                 <022>
12004 FOR I=0 TO 62:READ Q:POKE 50880+I,Q:NEXT
                          <222>
12005 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,32,0,0,120,0,0,254,0,1,255,128          <013>
12006 DATA 3,255,224,7,255,248,15,255,254,31,
   255,252,63,255,248,127,255,240,0,0        <184>
12007 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0         <024>
12010 FOR I=0 TO 62:READ Q:POKE 50944+I,Q:NEXT
                          <229>
12011 DATA 0,0,64,0,64,128,2,33,0,1,36,128,0,
   200,76,17,8,80,18,69,32,34,38,64,36       <209>
12012 DATA 100,48,20,162,64,8,164,128,4,153,192,
   5,22,40,2,20,16,5,38,0,8,169,0,4          <206>
12013 DATA 72,192,0,80,0,0,96,0,0,96,0,0,96,0
                          <020>
12015 FOR I=0 TO 62:READ Q:POKE 51008+I,Q:NEXT
                          <226>
12016 DATA 3,252,0,2,168,0,3,92,0,3,92,0,3,92,0,
   3,92,0,15,255,0,15,255,0,15,255           <136>
12017 DATA 0,15,255,208,15,255,208,15,252,0,14,
   168,0,7,252,0,7,252,0,3,252,0,3           <100>
12018 DATA 252,0,2,138,0,2,138,0,0,163,128,0,0,0
                          <153>
12020 FOR I=0 TO 62:READ Q:POKE 51072+I,Q:NEXT
                          <232>
12021 DATA 0,0,0,127,251,254,127,251,254,127,
   249,254,0,0,0,127,63,60,127               <079>
12022 DATA 127,190,127,127,190,62,127,190,0,0,0,
   127,231,254,127,239,254,127,239           <195>
12023 DATA 254,127,207,252,0,0,0,57,255,158,125,
   255,222,125,255,222,125,255,220           <185>
12024 DATA 0,0,0,255,255,255                 <233>
12026 FOR I=0 TO 62:READ Q:POKE 51136+I,Q:NEXT
                          <239>
12027 DATA 95,255,250,95,255,250,80,0,10,80,0,
   10,88,0,26,88,0,26,92,0,58,92,0,58        <244>
12028 DATA 94,0,122,94,0,122,95,0,250,95,0,250,
   95,129,250,95,126,250,94,255,122          <200>
12029 DATA 93,255,186,91,255,218,87,255,234,79,
   255,242,127,255,254,0,0,0                 <139>
12031 FOR I=0 TO 62:READ Q:POKE 51200+I,Q:NEXT
                          <236>
12032 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31,0,
   0,14,0,0,14,0,0,27,0,0,42,128,0           <034>
12033 DATA 113,192,0,106,192,0,59,128,0,21,0,0,
   14,0,0,0,0,0,0,0,0,0,0,0,0,0,0            <076>
12034 DATA 0,0,0,0                           <248>
12036 FOR I=0 TO 62:READ Q:POKE 51264+I,Q:NEXT
                          <252>
12038 DATA 0,0,0,0,0,0,0,0,1,85,128,2,170,64,
   2,170,0,2,175,0,2,173,0,2,174,192         <173>
12039 DATA 2,175,192,2,175,0,3,253,0,3,255,0,0,
   255,0,0,242,0,0,242,0,0,242,0,3           <086>
12040 DATA 252,0,3,252,0,0,0,0,0,0           <087>
12042 FOR I=0 TO 62:READ Q:POKE 51328+I,Q:NEXT
                          <003>
12043 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,
   16,6,0,30,57,120,31,192,112,16,6          <085>
12044 DATA 12,8,121,194,3,128,57,28,120,5,33,
   135,130,24,112,28,7,14,98,0,225,140       <245>
12045 DATA 0,30,48,0,1,192,0,0,0,0,0,0       <191>
12047 FOR I=0 TO 62:READ Q:POKE 51392+I,Q:NEXT
                          <009>
12048 DATA 0,0,0,15,255,240,10,0,80,10,0,80,10,
   0,80,19,255,200,20,0,40,20,0,40           <077>
12049 DATA 20,0,40,39,255,228,40,0,20,40,0,20,
   40,0,20,79,255,242,80,0,10,80,0,10        <201>
12050 DATA 80,0,10,159,255,249,160,0,5,160,0,5,
   255,255,255                               <203>
12052 FOR I=0 TO 62:READ Q:POKE 51456+I,Q:NEXT
                          <015>
12053 DATA 0,0,0,0,0,0,0,0,0,0,0,0,56,0,0,76,
   0,0,134,0,0,18,64,0,47,160,0,110          <094>
12054 DATA 160,0,191,64,0,156,64,0,56,0,0,68,0,
   1,132,0,2,8,0,5,20,0,0,0,0,0,0,0          <127>
12055 DATA 0,0,0,0,0,0                       <197>
12057 FOR I=0 TO 62:READ Q:POKE 51520+I,Q:NEXT
                          <012>
12058 DATA 16,0,0,21,0,0,31,64,0,31,80,0,31,213,
   0,31,254,64,31,255,64,31,255,64           <171>
12059 DATA 31,255,208,31,255,244,21,85,84,7,255,
   244,1,255,244,0,127,244,0,23,244          <020>
12060 DATA 0,5,244,0,1,244,0,0,116,0,0,20,0,0,4,
   0,0,0                                     <146>
12062 FOR I=0 TO 62:READ Q:POKE 51584+I,Q:NEXT
                          <027>
12063 DATA 0,0,16,0,0,80,0,1,240,0,7,208,0,7,
```

Listing »Grab des Pharao« (Fortsetzung)

```
   208,0,95,208,1,255,208,5,255,208          <087>
12064 DATA 31,255,208,31,255,208,21,85,80,31,
   255,64,31,253,0,23,212,0,31,208,0         <126>
12065 DATA 31,64,0,29,0,0,20,0,0,16,0,0,0,0,0,
   0,0                                       <049>
12067 FOR I=0 TO 62:READ Q:POKE 51648+I,Q:NEXT
                     <033>
12068 DATA 0,0,0,0,0,0,0,0,0,0,0,3,255,192,2,
   0,64,2,239,64,2,239,64,2,15,64,2          <153>
12069 DATA 240,64,2,247,64,2,247,64,2,7,64,2,
   224,64,2,239,64,2,239,64,2,0,64,2         <182>
12070 DATA 247,64,2,247,64,2,0,64,3,255,192
                     <007>
12072 FOR I=0 TO 62:READ Q:POKE 51712+I,Q:NEXT
                     <030>
12073 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,8,0,0,55,0,0,59,0,0         <103>
12074 DATA 213,192,0,191,128,0,204,192,0,191,
   128,0,255,192,0,183,128,0,255,192,0       <019>
12075 DATA 55,0,0,55,0,0,55,0                <063>
12077 FOR I=0 TO 62:READ Q:POKE 51776+I,Q:NEXT
                     <045>
12078 DATA 0,0,0,0,56,0,0,124,0,0,84,0,0,124,0,
   0,40,0,0,56,0,1,255,0,2,214,128,2         <187>
12079 DATA 238,128,2,214,128,3,57,128,0,254,0,0,
   254,0,0,124,0,0,108,0,0,108,0,0           <186>
12080 DATA 108,0,0,108,0,0,108,0,0,230,0     <074>
12082 FOR I=0 TO 62:READ Q:POKE 51840+I,Q:NEXT
                     <042>
12083 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,28,0,0,62,0,0         <099>
12084 DATA 28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0                     <096>
12086 FOR I=0 TO 62:READ Q:POKE 51904+I,Q:NEXT
                     <047>
12087 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,225,
   128,0,30,3,128,0,48,112,1,204,15,14       <180>
12088 DATA 6,0,192,0,0,51,193,192,12,14,56,0,48,
   6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0           <144>
12089 DATA 0,0,0,0,0,0                       <231>
12091 FOR I=0 TO 62:READ Q:POKE 51968+I,Q:NEXT
                     <062>
12092 DATA 0,0,0,0,0,0,0,0,0,0,0,0,21,84,0,74,
   148,1,42,100,4,169,164,18,166             <078>
12093 DATA 164,85,90,164,106,154,164,106,154,
   144,106,154,124,106,153,205,106,151       <058>
12094 DATA 240,85,80,60,0,0,13,0,0,0,0,0,0,0,0,
   0,0,0,0                                   <221>
12096 FOR I=0 TO 62:READ Q:POKE 52032+I,Q:NEXT
                     <050>
12097 DATA 255,255,255,128,36,1,128,60,1,128,66,
   1,128,126,1,128,129,1,128,255,1           <011>
12098 DATA 129,0,129,129,255,129,130,0,65,131,
   255,193,132,0,33,135,255,225,136          <217>
12099 DATA 0,17,143,255,241,144,0,9,159,255,249,
   160,0,5,191,255,253,192,0,3,255           <012>
12100 DATA 255,255                           <090>
12102 FOR I=0 TO 62:READ Q:POKE 52096+I,Q:NEXT
                     <066>
12103 DATA 0,0,0,0,28,0,0,3,128,0,0,64,0,1,128,
   0,14,0,0,48,0,0,64,0,0,48,0,0,12          <161>
12104 DATA 0,0,3,0,0,0,128,0,1,0,0,6,0,0,24,0,0,
   96,0,0,224,0,0,192,0,0,0,0,0,0            <084>
12105 DATA 0,0,0,0                           <063>
12107 FOR I=0 TO 62:READ Q:POKE 52160+I,Q:NEXT
                     <063>
12108 DATA 0,6,0,0,24,0,0,32,0,0,48,0,0,12,0,0,
   6,0,0,1,0,0,3,0,0,12,0,0,48,0,0,64        <229>
12109 DATA 0,0,96,0,0,24,0,0,6,0,0,24,0,0,96,0,
   0,224,0,0,192,0,0,0,0,0,0,0,0,0.          <187>
12111 FOR I=0 TO 62:READ Q:POKE 52224+I,Q:NEXT
                     <068>
12112 DATA 0,168,42,160,170,10,0,42,2,160,47,0,
   0,15,0,160,12,0,0,40,0,0,255,0,3          <196>
12113 DATA 60,192,163,60,240,3,252,0,160,60,0,0,
   42,240,160,42,252,0,85,12,0,85            <191>
12114 DATA 12,0,65,12,160,65,12,0,65,15,160,65,
   5,0,0,0                                   <038>
12116 FOR I=0 TO 62:READ Q:POKE 52288+I,Q:NEXT
                     <083>
12117 DATA 160,0,0,128,42,0,0,170,10,0,168,0,0,
   240,10,0,240,0,0,48,10,0,235,0,3          <190>
12118 DATA 60,192,3,60,202,15,60,192,0,60,202,0,
   60,0,3,168,0,15,168,10,48,85,0            <200>
12119 DATA 48,85,10,48,65,0,112,65,10,80,65,0,0,
   0,0                                       <165>
12121 FOR I=0 TO 62:READ Q:POKE 52352+I,Q:NEXT
                     <080>
12122 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0             <024>
12123 DATA 0,0,63,255,252,127,255,254,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0         <088>
12124 DATA 0,0,0,0,0                         <174>
12125 FOR I=0 TO 62:READ Q:POKE 52416+I,Q:NEXT
                     <085>
12126 DATA 0,0,0,3,192,0,7,192,0,7,128,0,7,128,
   0,2,128,0,7,0,0,59,128,0,47,192,0         <034>
12127 DATA 7,224,0,14,0,0,13,240,0,14,248,0,7,
   124,0,7,190,0,5,111,0,2,139,128,2         <229>
12128 DATA 130,192,7,129,64,31,255,252,31,255,
   252                                       <105>
12130 FOR I=0 TO 62:READ Q:POKE 52480+I,Q:NEXT
                     <091>
12131 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,2,170,128,10,170,160          <134>
12132 DATA 9,221,224,11,119,96,10,170,160,11,
   170,96,9,221,224,11,119,96,10,170         <240>
12133 DATA 160,8,0,32,8,0,32,8,0,32,0,0,0    <181>
12139 FOR I=0 TO 62:READ Q:POKE 52544+I,Q:NEXT
                     <101>
12140 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,8,0,1,214,0,2         <156>
12141 DATA 33,0,4,76,128,5,144,128,4,139,0,4,
   100,0,7,24,0,3,128,0,1,128,0,0,0,0        <243>
12142 DATA 0,0,0                             <008>
12144 FOR I=0 TO 62:READ Q:POKE 52608+I,Q:NEXT
                     <107>
12145 DATA 0,0,0,0,0,0,0,120,0,0,48,0,0,48,0,0,
   120,0,0,180,0,1,50,0,2,121,0,4            <078>
12146 DATA 120,128,8,252,64,8,212,64,9,182,64,5,
   122,128,3,119,0,3,183,0,3,203,0           <036>
12147 DATA 1,222,0,0,220,0,0,120,0,0,0,0     <119>
12149 FOR I=0 TO 62:READ Q:POKE 52672+I,Q:NEXT
                     <113>
12150 DATA 0,0,0,0,0,0,255,192,0,255,192,0,
   255,192,1,128,96,1,128,96,1,128,96        <037>
12151 DATA 3,0,48,3,0,48,3,0,48,6,0,24,6,0,24,6,
   0,24,12,0,12,12,0,12,12,0,12,15           <230>
12152 DATA 255,252,13,154,236,10,205,148,15,255,
   252                                       <224>
12154 FOR I=0 TO 62:READ Q:POKE 52736+I,Q:NEXT
                     <119>
12155 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,42,0,
   0,21,0,0,38,0,0,21,0,0,42,0,0,21          <193>
12156 DATA 0,0,170,128,0,85,64,0,170,128,0,85,
   64,0,170,128,0,85,64,0,0,0,0,0,0,0        <056>
12157 DATA 0,0,0,0,0                         <207>
12159 FOR I=0 TO 62:READ Q:POKE 52800+I,Q:NEXT
                     <116>
12160 DATA 0,0,0,0,0,0,0,127,128,0,125,128,0,
   118,128,0,239,192,0,237,192,0,254         <228>
12161 DATA 192,1,166,224,1,223,224,1,254,224,3,
   157,112,3,127,240,3,245,240,6,251         <098>
12162 DATA 184,7,55,24,7,255,248,0,0,0,0,0,0,0,
   0,0,0,0,0                                 <161>
12164 FOR I=0 TO 62:READ Q:POKE 52864+I,Q:NEXT
                     <131>
12165 DATA 0,24,0,0,90,0,0,126,0,0,126,0,0,126,
   0,0,60,0,0,60,0,0,60,0,0,24,0,1           <158>
12166 DATA 255,128,3,110,192,3,118,192,3,110,
   192,1,223,128,0,126,0,0,126,0,0,102       <078>
12167 DATA 0,0,102,0,0,102,0,1,255,128,7,255,224
                     <050>
15000 RETURN                                 <096>
19000 IF AP>7 THEN 19100                     <040>
19001 IF AP=7 THEN 19200                     <043>
19002 IF AP=6 THEN GOSUB 19035               <190>
19003 POKE 53270,PEEK(53270)OR 16:POKE 53283,9
   :POKE 53282,6                             <230>
19004 PRINT TAB(10)"{BLUE}...................."
                     <071>
19006 PRINT TAB(10)"...................."   <042>
19008 PRINT TAB(10)"...................."   <044>
19010 PRINT TAB(10)".....{GREY 3}LMNOP{LIG.BLUE}
   ...{GREY 3}LOP{LIG.BLUE}...."             <153>
19012 FOR T=1 TO 10:PRINT TAB(10)"{GREY 3}
```

Listing »Grab des Pharao« (Fortsetzung)

```
   KKKKKKKKKKKKKKKKKKKKK":NEXT                    <194>
19031 GOSUB 21050:RETURN                          <238>
19035 POKE V+21,PEEK(V+21)OR 128:POKE 50175,27
   :POKE V+14,145:POKE V+15,90:POKE V+46,9  <092>
19036 POKE V+23,PEEK(V+23)OR 128
   :POKE V+29,PEEK(V+29)OR 128:RETURN         <138>
19100 IF AP=15 OR AP=18 OR AP=19 OR AP=20 OR AP
   =21 OR AP=29 OR AP=46 OR AP=52 THEN 19110
                  <054>
19102 GOTO 19999                                  <134>
19110 PRINT TAB(12)"{BROWN}&%...........↑£"
                  <190>
19111 PRINT TAB(12)"..@%.......V£.."             <077>
19112 PRINT TAB(12)"..,.@%....V£,..."            <074>
19113 PRINT TAB(12)"I#R←,.1//0,.XH(J"            <048>
19114 PRINT TAB(12)".,.,=S-CCπYJ-,.."            <029>
19115 PRINT TAB(12)".,.,..T{2SPACE}T..-,.."
                  <162>
19116 PRINT TAB(12)"//0/0/T{2SPACE}T/1/1//"
                  <188>
19117 PRINT TAB(12)".,.,U£TG+T&±.-.."             <050>
19118 PRINT TAB(12)"..UT.+....2,&±.."            <213>
19120 PRINT TAB(12)"↑I.,+......Z.,&%"            <080>
19121 PRINT TAB(12)".,.+........2,.."            <228>
19122 PRINT TAB(12)".,+..........Z.."            <143>
19123 PRINT TAB(12)".+............2."             <234>
19124 PRINT TAB(12)"+..............2"             <235>
19129 IF AP=19 THEN 20015                         <218>
19002 GOSUB 21032:RETURN                          <209>
19130 GOSUB 21032:RETURN                          <081>
19200 POKE 53270,PEEK(53270)OR 16:POKE 53282,7
   :POKE 53283,9                                  <173>
19204 PRINT TAB(10)"{BROWN}ZZZZNZZZZNZZZZNZZZZZ"
                  <105>
19205 PRINT TAB(10)"ZZZZNZZZZNZZZZNZZZZZ"    <213>
19206 PRINT TAB(10)"££££T££££T££££T£££££"    <140>
19207 FOR T=1 TO 3:PRINT TAB(10)"ZZ0ZZZZ0ZZZZ0Z
   ZZZ0ZZ":NEXT                                   <093>
19210 PRINT TAB(10)"SSSSDSSS{3SPACE}SSSDSSSSS"
                  <233>
19211 PRINT TAB(10)"ZZZZNZZZ{3SPACE}ZZZNZZZZZ"
                  <215>
19212 PRINT TAB(10)"££££T£££{3SPACE}£££T£££££"
                  <152>
19213 FOR T=1 TO 4:PRINT TAB(10)"{GREY 3}
   KKKKKKKKKKKKKKKKKKKKK":NEXT                    <095>
19217 IF X6=1 THEN POKE V+21,PEEK(V+21)OR 128
   :POKE 50175,31:POKE V+14,168:POKE V+15,108
                  <070>
19218 POKE V+46,9                                 <195>
19219 GOSUB 21050                                 <227>
19220 RETURN                                      <237>
19999 POKE 53270,PEEK(53270)AND 239               <146>
20000 PRINT TAB(10)"{BROWN}&%...............↑£"
                  <210>
20001 PRINT TAB(10)"..&%...........↑£.."          <094>
20002 PRINT TAB(10)"....1///1///1//0...."         <037>
20003 PRINT TAB(10)"I#!←...-...-...,*)(J"         <079>
20004 PRINT TAB(10)".,..1/1///1///10.,.."         <037>
20005 PRINT TAB(10)".,..-.-...-...-,...."         <011>
20006 PRINT TAB(10)"////1///1///1//0////"         <049>
20007 PRINT TAB(10)"....-...-...-.,....."         <018>
20008 PRINT TAB(10)"..↑£1/1///1///10&%.."         <122>
20009 PRINT TAB(10)"↑I..-.-...-...-,.,&%"         <214>
20010 PRINT TAB(10)".,.+////////////2,.."         <035>
20011 PRINT TAB(10)".,+..............Z.."         <194>
20012 PRINT TAB(10)".+...............2."          <029>
20013 PRINT TAB(10)"+.................2{GREY 2}
   "                                              <182>
20014 IF AP=50 THEN PRINT"{BROWN,HOME,7DOWN,
   16RIGHT}T{6RIGHT}T{4RIGHT}T"                   <099>
20015 IF AP=19 OR AP=32 OR AP=55 THEN 20017
   :REM TREPPE                                    <164>
20016 GOTO 20020                                  <251>
20017 POKE V+21,PEEK(V+21)OR 128:POKE 50175,35
   :POKE V+14,160:POKE V+15,145                   <195>
20018 POKE V+29,PEEK(V+29)OR 128:POKE V+46,0
   :IF AP=19 THEN 19130                           <037>
20020 IF AP<>33 AND AP<>56 AND AP<>58 THEN 20900
                  <042>
20021 POKE V+21,PEEK(V+21)OR 128:POKE 50175,45
   :POKE V+14,159:POKE V+15,97                    <166>
20022 POKE V+46,0:POKE V+23,PEEK(V+23)OR 128
   :POKE V+29,PEEK(V+29)OR 129                    <056>
20023 PRINT"{BROWN,HOME,6DOWN,17RIGHT}......
   {DOWN,6LEFT}.+..2.{DOWN,6LEFT}+....2{DOWN,
   6LEFT}"                                        <216>
20024 PRINT"{UP,17RIGHT}......{DOWN,6LEFT}
   ......{DOWN,6LEFT}"                            <135>
20900 IF AP=10 THEN 21001                         <191>
21000 IF PN(AP,1)=0 THEN 21005                    <067>
21001 POKE 49363,55:POKE 49364,56:POKE 49403,52
   :POKE 49404,48:POKE 49443,54:POKE 49444,48
                  <185>
21002 POKE 49483,68:POKE 49484,44:POKE 49523,70
   :POKE 49524,57:POKE 49563,53:POKE 49564,44
                  <200>
21003 POKE 49603,27:POKE 49604,71:POKE 49643,68
   :POKE 49644,46:POKE 49683,51                   <083>
21005 IF PN(AP,2)=0 THEN 21010                    <069>
21006 POKE 49379,69:POKE 49380,65:POKE 49419,49
   :POKE 49420,29:POKE 49459,49:POKE 49460,54
                  <216>
21007 POKE 49499,45:POKE 49500,68:POKE 49539,64
   :POKE 49540,62:POKE 49579,45:POKE 49580,61
                  <211>
21008 POKE 49619,72:POKE 49620,60:POKE 49659,46
   :POKE 49660,68:POKE 49700,59                   <094>
21010 IF PN(AP,3)=0 THEN 21030                    <077>
21013 POKE 49410,36:POKE 49411,67:POKE 49412,67
   :POKE 49413,39:POKE 49449,48:POKE 49450,44
                  <192>
21014 POKE 49451,32:POKE 49452,32:POKE 49453,45
   :POKE 49454,49:POKE 49489,44:POKE 49490,44
                  <202>
21015 POKE 49491,32:POKE 49492,32:POKE 49493,45
   :POKE 49529,48:POKE 49530,44:POKE 49414,45
                  <201>
21016 POKE 49531,100:POKE 49532,103
   :POKE 49533,45:POKE 49534,49:POKE 49569,44
   :POKE 49570,43                                 <032>
21017 POKE 49571,46:POKE 49572,46:POKE 49573,50
   :POKE 49574,45:POKE 49610,46:POKE 49611,46
                  <204>
21018 POKE 49612,46:POKE 49613,46:POKE 49409,44
                  <124>
21030 Z1=0:IF IN(10,1)=AP OR IN(23,
   1)=AP OR IN(26,1)=AP OR IN(28,
   1)=AP THEN 21034                               <019>
21031 IF IN(35,1)=AP OR IN(36,1)=AP THEN 21034
                  <054>
21032 IF IN(13,1)=AP OR IN(17,1)=AP OR IN(34,
   1)=AP OR IN(42,1)=AP THEN Z1=1:GOTO 21036
                  <162>
21033 GOTO 21050                                  <252>
21034 PRINT"{HOME,7DOWN,18RIGHT,4SPACE,DOWN,
   4LEFT,4SPACE,DOWN,4LEFT,4SPACE,DOWN,4LEFT,
   4SPACE}"                                       <124>
21035 POKE V+10,161:POKE V+11,97
   :POKE V+29,PEEK(V+29)OR 32                     <003>
21036 POKE V+21,PEEK(V+21)OR 32:POKE 50173,39
   :POKE V+44,11:POKE V+23,PEEK(V+23)OR 32 <078>
21037 IF Z1=1 THEN POKE V+10,161:POKE V+11,78
   :POKE V+29,PEEK(V+29)OR 32                     <164>
21038 IF Z1=1 THEN PRINT"{HOME,4DOWN,18RIGHT,
   4SPACE,DOWN,4LEFT,4SPACE,DOWN,4LEFT,4SPACE,
   DOWN,4LEFT,4SPACE,DOWN,4LEFT,4SPACE}"          <114>
21050 IF IN(1,1)<>AP THEN 21060                   <034>
21051 POKE V+21,PEEK(V+21)OR 1:POKE 50168,26
   :POKE V+0,200:POKE V+1,120:POKE V+39,11 <190>
21052 POKE V+23,PEEK(V+23)OR 1
   :POKE V+29,PEEK(V+29)OR 1                      <214>
21060 IF IN(2,1)<>AP THEN 21070                   <046>
21061 POKE V+21,PEEK(V+21)OR 128:POKE 50175,29
   :POKE V+14,150:POKE V+15,110                   <213>
21062 POKE V+46,0:POKE V+37,9:POKE V+38,1
   :POKE V+23,PEEK(V+23)OR 128:POKE V+28,128
                  <085>
21063 POKE V+29,PEEK(V+29)OR 128                  <144>
21070 IF IN(3,1)<>AP THEN 21080                   <058>
21071 POKE V+21,PEEK(V+21)OR 64:POKE 50174,28
   :POKE V+12,160:POKE V+13,92                    <130>
21072 POKE V+45,15:POKE V+29,PEEK(V+29)OR 64
   :POKE V+23,PEEK(V+23)OR 64                     <040>
21080 IF IN(4,1)<>AP THEN 21090                   <070>
21081 POKE V+21,PEEK(V+21)OR 128:POKE 50175,30
```

Listing »Grab des Pharao« (Fortsetzung)

```
   :POKE V+14,168:POKE V+15,108                 <241>
21082 POKE V+46,9                               <018>
21090 IF IN(5,1)<>AP THEN 21100                 <073>
21091 IF IN(14,1)<>-1 AND IN(14,
   1)<>35 THEN 21095                            <247>
21092 POKE V+21,PEEK(V+21)OR 128:POKE 50175,53
   :POKE V+46,5:POKE V+14,195                   <161>
21093 POKE V+15,130:POKE V+29,PEEK(V+29)OR 128
   :GOTO 21100                                  <092>
21095 X8=1:POKE V+21,PEEK(V+21)OR 128
   :POKE 50175,46:POKE V+46,3                   <080>
21096 POKE V+14,190:POKE V+15,140:POKE V+29,128
   :GOTO 164                                    <213>
21100 IF IN(6,1)<>AP THEN GOTO 21110            <222>
21101 POKE V+21,PEEK(V+21)OR 8:POKE 50171,33
   :POKE V+28,8:POKE V+6,141:POKE V+7,103       <212>
21102 POKE V+42,11:POKE V+37,0:POKE V+38,15
                          <176>
21110 IF IN(7,1)<>AP THEN 21120                 <097>
21115 PRINT"{HOME,BROWN,8DOWN,15RIGHT},H{DOWN,
   2LEFT}10{DOWN,2LEFT}-,{DOWN,2LEFT}-,{DOWN,
   2LEFT}//"                                    <030>
21120 IF IN(9,1)<>AP THEN 21130                 <110>
21121 IF AP=19 OR AP=51 THEN 21130              <002>
21122 POKE V+21,PEEK(V+21)OR 2:POKE 50169,34
   :POKE V+2,150:POKE V+3,140                   <069>
21123 POKE V+29,PEEK(V+29)OR 2:POKE V+40,0      <244>
21130 IF IN(11,1)<>AP THEN 21140                <162>
21131 POKE V+21,PEEK(V+21)OR 64:POKE 50174,37
   :POKE V+12,235:POKE V+13,122                 <235>
21132 POKE V+28,64:POKE V+37,0:POKE V+38,9
   :POKE V+23,PEEK(V+23)OR 64                   <161>
21140 IF IN(12,1)<>AP THEN 21150                <174>
21141 POKE V+21,PEEK(V+21)OR 64:POKE 50174,38
   :POKE V+12,111:POKE V+13,119                 <245>
21142 POKE V+28,64:POKE V+37,0:POKE V+38,9
   :POKE V+23,PEEK(V+23)OR 64                   <171>
21150 IF IN(14,1)<>AP THEN 21160                <187>
21151 POKE V+21,PEEK(V+21)OR 4:POKE 50170,32
   :POKE V+4,170:POKE V+5,130                   <095>
21152 POKE V+41,12:POKE V+29,PEEK(V+29)OR 4
   :POKE V+23,PEEK(V+23)OR 4                    <005>
21160 IF IN(15,1)<>AP THEN 21170                <199>
21161 POKE V+21,PEEK(V+21)OR 128:POKE 50175,36
   :POKE V+14,170:POKE V+15,143                 <063>
21162 POKE V+46,0:POKE V+29,PEEK(V+29)OR 128
                          <138>
21170 IF IN(16,1)<>AP THEN 21180                <211>
21171 PRINT"{HOME,7DOWN}";TAB(24)"F"            <204>
21180 IF IN(18,1)<>AP AND IN(19,
   1)<>AP THEN 21200:REM SPRUENGE I.D. DECKE
                          <162>
21181 POKE V+21,PEEK(V+21)OR 128:POKE 50175,43
   :POKE V+14,170:POKE V+15,56                  <036>
21182 POKE V+46,0:POKE V+29,PEEK(V+29)OR 128
                          <158>
21200 IF IN(20,1)<>AP THEN 21210                <230>
21201 IF AP=31 THEN POKE 55599,8:GOTO 21210
                          <193>
21202 POKE V+21,PEEK(V+21)OR 8:POKE 50171,42
   :POKE V+6,180:POKE V+7,150:POKE V+42,8       <058>
21210 IF IN(21,1)<>AP THEN 21220                <242>
21211 POKE V+21,PEEK(V+21)OR 128
   :IF IN(46,3)=0 THEN POKE V+14,150
   :POKE V+15,121:GOTO 21213                    <132>
21212 POKE V+14,155:POKE V+15,115               <235>
21213 POKE 50175,44:POKE V+46,9:POKE V+37,0
   :POKE V+38,1:POKE V+23,PEEK(V+23)OR 128 <081>
21214 POKE V+29,PEEK(V+29)OR 128:POKE 49209,50
   :POKE 49212,50:POKE 49250,47:POKE V+28,128
                          <099>
21215 POKE 49251,47:POKE 49252,47:POKE 49210,45
                          <056>
21220 IF IN(22,1)<>AP THEN 21230                <254>
21221 POKE V+21,PEEK(V+21)OR 128:POKE 50175,41
                          <130>
21222 POKE V+14,190:POKE V+15,105:POKE V+45,14
   :POKE V+46,7                                 <092>
21223 POKE V+23,PEEK(V+23)OR 128
   :POKE V+29,PEEK(V+29)OR 128                  <085>
21230 IF IN(24,1)<>AP THEN 21240                <011>
21231 PRINT"{HOME,12DOWN,16RIGHT,GREY 1}
   1/1///10{DOWN,8LEFT}-.-...-,{DOWN,8LEFT}"
                          <064>
21232 PRINT"{UP,16RIGHT,BROWN}ROOOOOOE"         <206>
21240 IF IN(25,1)<>AP THEN 21250                <023>
21241 POKE V+21,PEEK(V+21)OR 192:POKE 50174,48
   :POKE 50175,49:POKE V+45,7:POKE V+46,7
   :POKE V+12,138                               <064>
21242 POKE V+37,0:POKE V+38,12:POKE V+13,98
   :POKE V+15,98:POKE V+28,192
   :POKE V+23,PEEK(V+23)OR 192                  <067>
21243 IF Y2=1 THEN POKE V+12,135:POKE V+14,208
   :POKE V+29,32:POKE 50173,31:POKE V+44,9
   :GOTO 21247                                  <055>
21244 PRINT"{HOME,6DOWN,14RIGHT,GREY 1}
   ............{DOWN,12LEFT}............"
   :POKE V+14,186                               <143>
21245 PRINT"{14RIGHT}............{DOWN,12LEFT}
   ............"                                <081>
21246 POKE V+29,PEEK(V+29)OR 192
   :PRINT"{14RIGHT}............":GOTO 21250
                          <218>
21247 PRINT"{HOME,6DOWN,14RIGHT,GREY 1}...
   {6SPACE}...{DOWN,12LEFT}...{6SPACE}..." <105>
21248 PRINT"{14RIGHT}...{6SPACE}...{DOWN,12LEFT}
   ...{6SPACE}..."                              <091>
21249 PRINT"{14RIGHT}...{6SPACE}..."            <219>
21250 IF IN(27,1)<>AP THEN 21260                <037>
21251 POKE V+21,PEEK(V+21)OR 128:POKE 50175,51
   :POKE V+46,12:POKE V+14,200                  <096>
21252 POKE V+15,120:POKE V+23,PEEK(V+23)OR 128
   :POKE V+29,PEEK(V+29)OR 128                  <105>
21260 IF IN(29,1)<>AP THEN 21270                <050>
21261 POKE V+21,PEEK(V+21)OR 192:POKE 50174,52
   :POKE 50175,54                               <021>
21262 POKE V+12,121:POKE V+13,105:POKE V+14,210
   :POKE V+15,105:POKE V+28,64                  <209>
21264 POKE V+45,8:POKE V+46,15:POKE V+37,11
   :POKE V+38,7                                 <252>
21265 POKE V+23,PEEK(V+23)OR 192
   :POKE V+29,PEEK(V+29)OR 64                   <080>
21270 IF IN(30,1)<>AP THEN 21280                <053>
21272 POKE V+21,PEEK(V+21)OR 24:POKE 50171,50
   :POKE 50172,50:POKE V+6,142:POKE V+7,140
                          <113>
21273 POKE V+8,172:POKE V+9,140:POKE V+42,8
   :POKE V+43,8:POKE V+29,PEEK(V+29)OR 24  <255>
21280 IF IN(33,1)<>AP THEN 21290                <035>
21281 PRINT"{BROWN,HOME,DOWN,16RIGHT}2-....,+
   {DOWN,7LEFT}//////"                          <190>
21290 IF IN(37,1)<>AP THEN 21300                <073>
21291 POKE V+21,PEEK(V+21)OR 128:POKE 50175,58
   :POKE V+46,12:POKE V+14,138                  <153>
21292 POKE V+15,105:POKE V+23,PEEK(V+23)OR 128
   :POKE V+29,PEEK(V+29)OR 128                  <148>
21300 IF IN(38,1)<>AP THEN 21310                <085>
21301 POKE V+21,PEEK(V+21)OR 208:POKE V+28,64
   :POKE 50175,55:POKE 50172,57                 <016>
21302 POKE 50174,56:POKE V+12,160:POKE V+13,125
   :POKE V+14,160:POKE V+15,125:POKE V+46,12
                          <102>
21303 POKE V+9,125:POKE V+45,9:POKE V+37,0 <084>
21304 POKE V+43,12:POKE V+23,208
   :POKE V+29,PEEK(V+29)OR 208                  <068>
21305 POKE V+8,160:IF IN(45,
   1)=-1 OR Y3=1 THEN POKE V+8,180              <121>
21307 PRINT"{HOME,10DOWN,19RIGHT,2SPACE,DOWN,
   2LEFT,2SPACE,DOWN,2LEFT,2SPACE,DOWN,3LEFT,
   4SPACE}"                                     <205>
21310 IF IN(39,1)<>AP THEN 21320                <097>
21311 POKE V+21,PEEK(V+21)OR 128:POKE 50175,30
   :POKE V+46,9:POKE V+14,172                   <119>
21312 POKE V+15,98:POKE V+23,PEEK(V+23)OR 128
                          <082>
21314 PRINT"{HOME,6DOWN,18RIGHT,BLACK}1..0{DOWN,
   4LEFT}-..,{DOWN,4LEFT}-..,"                  <182>
21315 PRINT"{18RIGHT}-..,{DOWN,4LEFT}-..," <108>
21318 PRINT"{HOME,14DOWN,16RIGHT,BROWN}+1////02"
                          <086>
21320 IF IN(40,1)<>AP THEN 21330                <100>
21321 POKE V+21,PEEK(V+21)OR 128:POKE 50175,52
   :POKE V+28,128:POKE V+46,8:POKE V+37,11 <083>
21322 POKE V+38,7:POKE V+14,160:POKE V+15,110
   :POKE V+23,PEEK(V+23)OR 128                  <021>
21323 POKE V+29,PEEK(V+29)OR 128                <149>
```

Listing »Grab des Pharao« (Fortsetzung)

```
21330 IF IN(44,1)<>AP THEN 21340                <115>
21331 PRINT"{HOME,12DOWN,15RIGHT,BROWN}
   "CHR$(176)                                    <096>
21340 IF IN(45,1)<>AP THEN 21350                <127>
21341 POKE V+21,PEEK(V+21)OR 32
   :POKE V+28,PEEK(V+28)OR 32:POKE V+44,14
   :POKE V+37,0                                  <187>
21342 POKE V+38,7:POKE V+29,PEEK(V+29)OR 32
   :POKE V+10,160:POKE V+11,122:POKE 50173,40
                    <091>
21350 IF IN(46,1)<>AP THEN 21370:REM GASMASKE
                    <161>
21360 PRINT"{HOME,12DOWN,19RIGHT,BROWN}R"       <239>
21370 IF IN(41,1)<>AP THEN 21500                <150>
21372 IF AP=57 THEN 21500                       <168>
21375 PRINT"{HOME,13DOWN,20RIGHT,BROWN}T"       <046>
21500 RETURN                                    <221>
30000 POKE V+21,0:PRINT"{CLR,3DOWN,10RIGHT,
   GREEN}SIE TRAGEN BEI SICH"                   <177>
30010 FOR I=1 TO 46                             <106>
30020 IF IN(I,1)=-1 THEN PRINT"{BLUE,DOWN,
   10RIGHT,WHITE}"IT$(I)                        <150>
30025 NEXT                                      <064>
30026 PRINT"{5DOWN,10RIGHT,GREEN,RVSON}SPACE
   DRUECKEN{RVOFF,WHITE}"                       <151>
30030 GET A$:IF A$<>" "THEN 30030               <056>
30040 IF A$=" "THEN GOTO 160                    <122>
34000 POKE V+21,0:PRINT CHR$(14),"{CLR,3DOWN}"
   :INPUT"FILENAME";E$                          <152>
34010 OPEN 2,1,2,E$                             <127>
34020 FOR I=1 TO 58                             <038>
34030 PRINT#2,PN(I,1):PRINT#2,PN(I,2)
   :PRINT#2,PN(I,3):PRINT#2,PN(I,4)             <082>
34035 PRINT#2,PN(I,5):PRINT#2,PN(I,6):NEXT      <140>
34040 FOR I=1 TO 46                             <055>
34050 PRINT#2,IN(I,1):PRINT#2,IN(I,2)
   :PRINT#2,IN(I,3):NEXT                        <070>
34060 PRINT#2,AP:PRINT#2,X1:PRINT#2,X2
   :PRINT#2,X4:PRINT#2,X5:PRINT#2,X6            <194>
34070 PRINT#2,X9:PRINT#2,Y1:PRINT#2,Y2
   :PRINT#2,Y3:PRINT#2,Y4:PRINT#2,GE            <204>
34080 CLOSE 2:GOTO 50300                        <018>
35000 PRINT"{CLR,3DOWN}":INPUT"FILENAME";E$:KK=1
                    <254>
35005 OPEN 2,1,0,E$                             <100>
35010 FOR I=1 TO 58                             <008>
35020 INPUT#2,PN(I,1),PN(I,2),PN(I,3),PN(I,4),
   PN(I,5),PN(I,6):NEXT                         <091>
35030 FOR I=1 TO 46                             <025>
35040 INPUT#2,IN(I,1),IN(I,2),IN(I,3):NEXT      <011>
35050 INPUT#2,AP,X1,X2,X4,X5,X6,X9,Y1,Y2,Y3,Y4,
   GE                                           <206>
35060 CLOSE 2:RETURN                            <022>
39900 DATA GEHE,ZERSTOERE,OEFFNE,KLOPFE,FRAGE,
   HACKE,TOETE,KRIECHE,WERFE,SCHIEBE            <127>
39910 DATA DREHE,KLETTERE,NIMM,VERLIERE,ZIEHE,
   LIST,SAVE,STOP                               <181>
39920 DATA SPITZHACKE,BEDUINE,STRAUCH,
   ZUGEMAUERTER EINGANG,SCHLANGE,STATUE,HEBEL
                    <054>
39925 DATA TUER NACH WESTEN,SEIL,
   TUER NACH NORDEN,DURCHGANG,DURCHGANG         <102>
39930 DATA TUER NACH NORDEN,TONKRUG MIT ESSEN,
   SKORPION,PFEIL,TUER NACH NORDEN              <127>
39935 DATA BESCHAEDIGTE DECKE,
   SPRUENGE IN DER DECKE,DIAMANT,
   TOTER GRABRAEUBER                            <058>
39940 DATA STATUE,TUER NACH NORDEN,LOCH,RELIEF,
   TUER NACH NORDEN,HORUS                       <131>
39945 DATA TUER NACH NORDEN,SCHAETZE,HOLZBALKEN,
   TUER NACH WESTEN,TUER                        <169>
39950 DATA OEFFNUNG IN DER DECKE,
   TUER NACH NORDEN,TUER NACH NORDEN            <235>
39955 DATA TUER NACH NORDEN,OSIRISSTATUE,
   SARKOPHAG,ZUGEMAUERTER AUSGANG               <203>
39960 DATA HOLZSCHREIN,AMULETT,TUER NACH NORDEN,
   TUER NACH SUEDEN,SCHLUESSEL                  <133>
39970 DATA TOTENMASKE,GASMASKE,GIFTIGE GASE,
   FALLTUER,SCHRITTE,SONNENAUFGANG              <018>
39975 DATA GEHEIMGANG,AUGE,WAND,TREPPE          <178>
39985 DATA 160,0,162,0,142,32,208,142,33,208,
   232,234,224,16,208,244,200,192,255           <082>
39990 DATA 208,237,169,0,141,32,208,141,33,208,
   96                                           <173>
40000 DATA 3,4,2,0,0,0,2,2,2,1,0,0,3,1,3,0,0,0,
   1,0,5,0,0,0,0,0,0,4,0,0                      <028>
40001 DATA 5,0,7,0,0,0,0,0,0,6,0,0,9,17,15,0,0,
   0,10,8,0,0,0,0,0,9,0,0,0,0                   <199>
40002 DATA 0,10,12,0,0,0,0,13,0,11,0,0,12,0,0,0,
   0,0,0,0,20,0,0,0,0,0,0,0,8,0,0               <052>
40003 DATA 0,0,0,15,0,0,8,0,18,0,0,0,0,0,19,17,
   0,0,0,0,0,18,0,58,0,0,21,14,0,0              <197>
40004 DATA 0,0,0,20,0,0,27,23,0,21,0,0,22,24,0,
   26,0,0,23,0,0,0,0,0,26,0,24,0,0,0            <023>
40005 DATA 0,25,23,0,0,0                        <204>
40006 DATA 0,22,0,28,0,0,31,0,27,29,0,0,0,0,28,
   0,0,0,0,0,0,0,0,0,28,0,30,0,0                <190>
40007 DATA 0,0,0,31,0,33,34,38,0,36,32,0,0,33,0,
   35,0,0,55,36,34,0,0,0,35,37,33,0             <084>
40008 DATA 0,0,36,41,0,0,0,0,33,0,0,0,0,0,0,40,
   46,0,0,0,39,0,0,0,0,0,37,42,0,0,0            <029>
40009 DATA 0,41,0,0,0,0,0,0,44,48,0,0,0,43,50,
   49,0,0,0,49,0,0,0,0,0,0,0,51,39              <169>
40010 DATA 0,0,0,0,0,40,0,0,0,49,0,43,0,0,48,45,
   0,44,0,0,44,0,0,38,0,0,0,0,0,46,0            <099>
40011 DATA 0,0,0,0,47,0,0,0,0,0,52,0,0,0,0,0,51,
   0,0,0,35,0,0,0,56,0,57,0,0,55,0              <243>
40012 DATA 56,0,0,0,0,0,0,0,0,19,0              <004>
40025 DATA 4,1,1,5,0,1,7,0,1                    <156>
40027 DATA 7,0,0,35,0,1,10,1,1                  <253>
40029 DATA 10,0,0,10,0,1,12,1,1                 <036>
40030 DATA 13,0,1,0,0,0                         <174>
40031 DATA 14,0,1,15,0,1,16,1,1,16,0,0          <127>
40032 DATA 21,0,0                               <154>
40033 DATA 21,0,1,23,0,1                        <230>
40034 DATA 24,0,1,25,1,1                        <237>
40035 DATA 27,1,1,31,0,1,31,0,1                 <055>
40036 DATA 37,0,1                               <166>
40037 DATA 40,0,1,42,0,1,42,0,1                 <055>
40038 DATA 38,0,1,47,0,0,48                     <144>
40039 DATA 1,1,50,0,1,50,0,0                    <167>
40040 DATA 51,0,1,52,0,1,0,0,0                  <006>
40041 DATA 47,0,1,52,0,1,53,0,1                 <069>
40043 DATA 54,0,1,57,0,1,0,1,1                  <019>
40044 DATA 19,0,1,25,0,1,58,1,1,53,1,0,27,1,0
                    <232>
45000 POKE 53270,PEEK(53270)OR 16:POKE 53283,9
   :POKE 53282,7                                <217>
45005 POKE 53280,0:POKE 53281,9                 <170>
45010 PRINT"{CLR,GREY 1,2DOWN,3RIGHT,4SPACE}P
   H A R A O S{3SPACE}G R A V E"                <132>
45020 PRINT"{DOWN,11SPACE}A D V E N T U R E
   {13SPACE,BLACK}"                             <183>
45030 PRINT"{19SPACE}.."                        <172>
45031 PRINT"{18SPACE}.//."                      <011>
45032 PRINT"{17SPACE}.////."                    <106>
45033 PRINT"{16SPACE}.//////."                  <201>
45034 PRINT"{15SPACE}.////////."                <040>
45035 PRINT"{14SPACE}.//////////."              <135>
45036 PRINT"{13SPACE}.////////////."            <230>
45037 PRINT"{12SPACE}.//////////////."          <069>
45038 PRINT"{11SPACE}.////////////////."        <164>
45039 PRINT"{10SPACE}.//////////////////."      <003>
45040 PRINT"{9SPACE}.////////////////////."     <098>
45041 PRINT"{8SPACE}.//////////////////////."
                    <193>
45050 PRINT"{2DOWN,GREY 1,19SPACE}BY{19SPACE}"
                    <218>
45060 PRINT"{12SPACE}WOLFGANG RAUSCH{12SPACE}"
                    <172>
45100 GOSUB 12000:GOSUB 30                      <134>
45110 PRINT"{CLR}",CHR$(14):POKE 53270,
   PEEK(53270)AND 239                           <094>
45112 PRINT"{WHITE,3DOWN,4RIGHT}N
   : NEUES SPIEL BEGINNEN"                      <188>
45113 PRINT"{DOWN,4RIGHT}A : ALTES SPIEL FORTSE
   TZEN"                                        <080>
45114 GET X$:IF X$="N"THEN DA=1:GOTO 45120      <090>
45116 IF X$="A"THEN GOSUB 35000:GOSUB 60
   :GOTO 45120                                  <066>
45118 IF X$<>"A"AND X$<>"N"THEN 45114           <165>
45120 PRINT"{CLR,RIGHT,YELLOW}ZIEL DES SPIELS
   IST ES EINE PYRAMIDE"                        <127>
45125 PRINT"{RIGHT}ZU ERFORSCHEN UND DIE WERTVO
   LLE"                                         <010>
45127 PRINT"{RIGHT}TOTENMASKE DES PHARAOS ZU
```

```
   FINDEN."                                       <103>
45130 PRINT"{DOWN,RIGHT,ORANGE}FOLGENDE VERBEN
   VERSTEHT DER COMPUTER:"                        <005>
45132 PRINT"{RIGHT,DOWN}GEHE,NIMM,VERLIERE,
   OEFFNE,TOETE"                                  <147>
45134 PRINT"{RIGHT}ZERSTOERE,KLOPFE,FRAGE,WERFE,
   DREHE"                                         <114>
45136 PRINT"{RIGHT}HACKE,KRIECHE,WERFE,SCHIEBE,
   KLETTERE"                                      <205>
45137 PRINT"{RIGHT}LIST,SAVE,STOP                 <226>
45140 PRINT"{BLACK,DOWN,RIGHT}
   NACH JEDEM EINGEGEBENEN WORT MUSS"             <014>
45142 PRINT"{RIGHT}'RETURN' GEDRUECKT WERDEN."
                    <191>
45144 PRINT"{RIGHT}ALLE NOMEN KOENNEN AUF 5,
   ALLE VERBEN"                                   <072>
45146 PRINT"{RIGHT}AUF 4 BUCHSTABEN ABGEKUERZT
   WERDEN."                                       <090>
45148 PRINT"{RIGHT}BEI RICHTUNGSANGABEN GENUEGT
   EIN "                                          <100>
45150 PRINT"{RIGHT}BUCHSTABE.                     <037>
45152 PRINT"{RIGHT}Z.B.:GEHE 'RETURN'{5SPACE}N
   'RETURN'"                                      <255>
45154 PRINT"{RIGHT}LIST ZEIGT ALLE GEGENSTAENDE,
   DIE"                                           <209>
45155 PRINT"{RIGHT}MAN BEI SICH TRAEGT (MAXIMAL
   3)."                                           <098>
45157 PRINT"{RIGHT}SAVE SPEICHERT DEN SPIELSTAN
   D AUF"                                         <015>
45158 PRINT"{RIGHT}KASSETTE AB."                  <165>
45160 IF DA=1 THEN GOSUB 60                       <165>
45170 PRINT"{DOWN,4RIGHT,RVSON}'SPACE' DRUECKEN
   {RVOFF}"                                       <179>
47000 GET X$:IF X$<>" "THEN 47000                 <247>
47010 IF X$=" "THEN PRINT CHR$(142),"{CLR}"
                    <072>
47020 IF KK=1 THEN 155                            <169>
47025 GOTO 150                                    <135>
49000 GOSUB 5600:PRINT"{CLR,2DOWN,9RIGHT}SIE
   SIND AUSSERHALB":GOSUB 5800                    <221>
49002 IF IN(45,1)<>-1 THEN 49050                  <075>
49005 PRINT CHR$(14),"{CLR,GREEN,3DOWN,3SPACE}
   SIE HABEN ES GESCHAFFT DIE GOLDENE"            <072>
49010 PRINT"{DOWN.3SPACE}TOTENMASKE DES PHARAOS
   ZU FINDEN"                                     <103>
49020 PRINT"{DOWN,3SPACE}UND HOECHSTE EHRUNGEN
   SIND IHNEN{3SPACE}"                            <223>
49030 PRINT"{DOWN,3SPACE}GEWISS,"                 <049>
49040 PRINT"{3DOWN,RED,3SPACE}DOCH DER FLUCH
   DES PHARAOS WIRD"                              <006>
49045 PRINT"{DOWN,3SPACE}SIE FUER IMMER VERFOLG
   EN.{WHITE}":GOTO 50300                         <101>
49050 PRINT"{CLR,GREEN,3DOWN,3SPACE}SIE HABEN
   ES ZWAR GESCHAFFT, DIE"                        <062>
49055 PRINT"{DOWN,3SPACE}PYRAMIDE LEBEND ZU
   VERLASSEN, DOCH"                               <125>
49060 PRINT"{DOWN,3SPACE}SIE HABEN DIE GOLDENE
   TOTENMASKE"                                    <219>
49065 PRINT"{DOWN,3SPACE}NICHT GEFUNDEN."
   :GOTO 50300                                    <001>
50100 REM                                         <006>
50105 POKE V+14,190:POKE V+15,130                 <053>
50110 FOR F2=135 TO 155 STEP 2:POKE 50175,46
   :FOR T=1 TO 100:NEXT:POKE V+15,F2              <006>
50120 POKE 50175,47:FOR T=1 TO 100:NEXT:NEXT
                    <090>
50121 FOR T=1 TO 30:SYS 866:NEXT:GOTO 50210
                    <062>
50190 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}SIE
   HABEN SICH HOFFNUNGSLOS VERIRRT{WHITE}"
   :GOTO 50300                                    <237>
50200 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}SIE
   HABEN DEN HALT VERLOREN UND SIND{4SPACE}
   HINUNTERGEFALLEN{WHITE}"                       <120>
50201 GOTO 50300                                  <094>
50202 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}SIE
   SIND IN EINE FALLGRUBE GESTUERZT{WHITE}"
   :GOSUB 5600:GOTO 50300                         <131>
50203 GOSUB 5600:PRINT CHR$(14)"{CLR,RED,2DOWN,
   3SPACE}DIE "IT$(54)" IST EINGESTUERZT{WHITE}"
   :GOTO 50300                                    <134>
50204 GOSUB 5600:PRINT CHR$(14)"{RED,CLR,2DOWN,
   3SPACE}DIE DECKE IST HERUNTERGEBROCHEN{WHITE}
   ":GOTO 50300                                   <117>
50206 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}EIN
   HERABFALLENDER STEIN HAT SIE ER{5SPACE}
   SCHLAGEN{WHITE}":GOTO 50300                    <181>
50207 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}
   "IT$(47)" HABEN SIE GETOETET{WHITE}"
   :GOTO 50300                                    <047>
50208 FOR T=1 TO 20:SYS 866:NEXT:GOSUB 5600
   :PRINT"{CLR,RED,2DOWN,3SPACE}SIE WURDEN VON
   HINTEN GEPACKT UND "                           <137>
50209 PRINT"{3SPACE}ERWUERGT{WHITE}":GOTO 50300
                    <247>
50210 GOSUB 5600:PRINT CHR$(14)"{CLR,RED,2DOWN,
   3SPACE}DIE "IT$(5)" HAT SIE GEBISSEN{WHITE}"
   :GOTO 50300                                    <206>
50211 GOSUB 5600:PRINT CHR$(14)"{CLR,RED,2DOWN,
   3SPACE}EINE "IT$(48)" HAT SICH UNTER IHNEN"
                    <139>
50212 PRINT"{3SPACE}GEOEFFNET{WHITE}":GOTO 50300
                    <024>
50213 GOSUB 5600:PRINT CHR$(14)"{CLR,RED,2DOWN,
   3SPACE}SIE HABEN EINEN MECHANISMUS "           <145>
50214 PRINT"{3SPACE}AUSGELOEST ,WODURCH EIN
   "IT$(16)" AUS "                                <141>
50215 PRINT"{3SPACE}DER WAND GESCHOSSEN WURDE
   {WHITE}":GOTO 50300                            <006>
50300 PRINT CHR$(14)"{5DOWN,3RIGHT}NOCH EIN
   VERSUCH? (J/N)"                                <097>
50310 GET A$:IF A$<>"J"AND A$<>"N"THEN 50310
                    <022>
50320 IF A$="J"THEN CLR:PRINT CHR$(142)"{CLR}"
   :RUN                                           <228>
50330 SYS 64738                                   <008>
50700 FOR X=1 TO 20:POKE V+8,160+X
   :FOR T=1 TO 200:NEXT:NEXT:IN(45,3)=1
   :GOSUB 5900:GOTO 164                           <194>
50720 TI$="000000"                                <157>
50725 PRINT A4$"KOMMANDO ?"                       <246>
50730 POKE 631,0:POKE 198,0                       <076>
50731 U=PEEK(631)                                 <165>
50740 IF TI$="000002"THEN 50770                   <008>
50750 IF U=71 THEN POKE 631,U:GOTO 250            <177>
50760 IF U>0 AND U<>71 THEN 50770                 <073>
50765 GOTO 50731                                  <156>
50770 FOR X=1 TO 10:POKE V+13,125-X
   :FOR T=1 TO 300:NEXT:NEXT:FOR T=1 TO 20
   :SYS 866:NEXT                                  <166>
50780 GOSUB 5600:PRINT CHR$(14)"{CLR}"
   :GOTO 50300                                    <092>
50800 TI$="000000":POKE 631,0                     <212>
50805 PRINT A4$"KOMMANDO ?"                       <070>
50806 POKE 631,0:POKE 198,0                       <152>
50807 U=PEEK(631)                                 <241>
50808 IF U=71 OR U=84 THEN POKE 631,U:GOTO 250
                    <014>
50809 IF U>0 AND U<>71 AND U<>84 THEN 50820
                    <074>
50810 IF TI$="000002"THEN 50820                   <074>
50815 GOTO 50807                                  <210>
50820 FOR X=1 TO 10:POKE V+15,140+X
   :POKE V+14,170-X/2:FOR T=1 TO 100:NEXT:NEXT
                    <128>
50825 FOR T=1 TO 10:SYS 866:NEXT:GOSUB 5600
                    <214>
50830 PRINT CHR$(14)"{CLR,RED,2DOWN,3SPACE}DER
   "IT$(15)" HAT SIE GESTOCHEN{WHITE}"
   :GOTO 50300                                    <052>
52000 PRINT A3$"{GREEN}SIE HOEREN "IT$(49)"
   HINTER SICH{WHITE}":C=C+1                      <006>
52004 FOR C1=1 TO 4                               <159>
52005 FOR LA=5 TO 0 STEP-.4                       <111>
52010 POKE SI+24,LA:POKE SI+5,8*16+10
   :POKE SI+6,15*16:POKE SI+1,15:POKE SI+0,0
                    <046>
52020 POKE SI+4,129:NEXT                          <148>
52025 POKE SI+4,0:POKE SI+5,0                     <025>
52030 FOR T=1 TO 800:NEXT:NEXT:RETURN             <061>
```

Listing »Grab des Pharao« (Schluß)

Checksummer — keine Fehler mehr beim Abtippen von Listings

Das Programm Checksummer wird Ihnen das Abtippen von Listings erheblich erleichtern. Es wird deshalb in dieser und in der nächsten Ausgabe noch einmal mit ausführlicher Beschreibung abgedruckt. Lesen Sie bitte die Anleitung genau durch, bevor Sie ein Listing mit dem Checksummer eintippen.

Der Checksummer 64 ist ein kleines Maschinenprogramm, das, wenn es aktiviert ist, Sie sofort davon unterrichtet, ob Sie die jeweilige Programmzeile korrekt eingegeben haben.

1. Tippen Sie den Basic-Lader sorgfältig ein. Es gibt zwei Versionen: eine für den Commodore 64 und eine für den VC 20.
2. Bevor Sie »RUN« eingeben, speichern Sie den Basic-Lader bitte erst ab, denn wenn Sie zum Beispiel einen Fehler bei den eingetippten POKE-Anweisungen gemacht haben, ist es möglich, daß der Rechner aussteigt. Heben Sie sich den abgespeicherten Checksummer 64 auf — Sie werden ihn immer wieder brauchen, wenn Sie ein Basic-Programm aus dem 64'er eintippen wollen.
3. Der Checksummer 64 überprüft sich selbst. Wenn Sie einen Fehler in den DATAs gemacht haben, listen Sie die fehlerhafte Zeile einfach, korrigieren sie und starten dann das Programm neu.
4. Nach Initialisierung des Maschinenprogramms ist der Checksummer 64 aktiviert. Er steht innerhalb des Betriebssystems und verbraucht kein einziges Byte Speicherplatz. Es sei hier für Interessierte gesagt, daß selbst alle Sprungvektoren unverändert bleiben, das Programm also mit einer Vielzahl von anderen Programmier-Spracherweiterungen wie etwa Exbasic Level II problemlos zusammenarbeitet. Achten Sie aber darauf, daß bestimmte Spracherweiterungen das hinter dem ROM liegende RAM für Hires-Grafiken benutzen. Wird zum Beispiel eine Hires-Grafik von Simons Basic aus angesprochen, so wird der Checksummer 64 zerstört.
5. Wenn Sie den Checksummer 64 zwischenzeitlich nicht benutzen, können Sie ihn jederzeit mit »POKE 1, 55« desaktivieren. Auch durch Drücken der Run-Stop- und der Restore-Taste wird der Checksummer 64 desaktiviert. Wollen Sie, daß der Checksummer 64 auch noch nach Drücken dieser Tastenkombination erhalten bleibt, so geben Sie bei aktiviertem Checksummer 64 »POKE 64982, 53« ein. Der Checksummer 64 ist dann nur durch »POKE 1, 55« abschaltbar.
   Wollen Sie den Checksummer 64 wieder einschalten, so geben Sie bitte »POKE 1, 53« ein.
   Das Maschinenprogramm bleibt solange erhalten, bis der Computer ausgeschaltet, oder wenn von anderen Programmen auf das hinter dem ROM liegende RAM zugegriffen wird.
6. Eine Checksumme wird nur dann ausgegeben, wenn der Commodore 64 (VC 20) eindeutig erkennt, daß Sie eine Zeile bestehend aus der Zeilennummer und zumindestens einem alphanumerischen Zeichen eingegeben haben. Ansonsten reagiert der Commodore 64 normal.

Hinweis: Wenn Sie bei aktiviertem Checksummer 64 ein Programm mit »LOAD« in den Speicher holen, wird auch eine Checksumme ausgegeben. Dies liegt jedoch an rechnerinternen Routinen und hat keine weitere Bedeutung, stellt insbesondere keine Gefahr für das geladene Programm dar, da alle Pointer richtig gesetzt werden.

Nach Eingabe von RUN wird zunächst einmal das ROM in das RAM des Commodore 64 verschoben, wonach der Basic-Interpreter modifiziert wird. Dadurch hat man den Vorteil, trotz einer zusätzlichen Routine das gesamte RAM des Rechners zur Verfügung zu haben. Nach ordnungsgemässem Ablauf des Programms können Sie sofort mit Eingaben beginnen. Für Maschinensprache-Spezialisten weise ich darauf hin, daß ich ausnutze, daß die Einschaltmeldungen des Rechners nur nach einem Reset generiert wird. Der Textbereich, in dem die Meldung steht, wird von dem erzeugten Maschinenprogramm überschrieben.

Alle veröffentlichten Listings sind mit einer Checksumme versehen, die am Ende jeder Programmzeile steht. Diese Checksumme steht zwischen < und >. Sie wird beim Eintippen des Programms nicht mit eingegeben. Die Zahl zwischen den beiden Zeichen stellt lediglich eine Information für Sie dar. Wenn Sie diese Checksumme dennoch mit eintippen, werden Sie schnell bemerken, daß Sie etwas falsch gemacht haben. Bei aktiviertem Checksummer 64 wird nämlich nach Eingabe einer Basic-Zeile, die mit Return beendet wird, in die linke obere Bildschirmecke die Checksumme eingeblendet, die mit der Summe aus dem veröffentlichten Listing übereinstimmen muß. Ist das nicht der Fall, haben Sie die Zeile anders eingegeben, als sie im Listing dargestellt ist. Vergessen Sie also bitte nicht, daß die am Ende einer Zeile in < und > stehende Prüfsumme nicht mit eingegeben werden darf.

Der Checksummer 64 ist so ausgelegt, daß er abhängig von der Zeilennummer und dem Text der Zeile eine Checksumme ausgibt. Beim Bilden dieser Checksumme werden Spaces (Leertaste) überlesen, was für Sie bedeutet, daß es egal ist, wieviel Leerzeichen Sie zwischen den Worten lassen, da Sie für den Programmablauf ohnehin keine Bedeutung haben. Aber manchmal ist das richtige Setzen von Leerzeichen doch wichtig, besonders innerhalb von Strings (Zeichenketten), die gedruckt werden sollen. Seien Sie deshalb besonders genau bei Leerzeichen, die innerhalb von Anführungszeichen stehen, denn meistens ermöglichen nur die richtig gesetzten Spaces eine sinnvolle Textausgabe auf dem Bildschirm.

Beachten Sie auch, daß es durchaus erlaubt ist, Abkürzungen für die Commodore-Befehlswörter zu verwenden. So führt die Eingabe von »?« als Kurzschreibweise für »PRINT« nicht etwa zu einem Checksummen-Fehler, sondern wird korrekt verarbeitet und dementsprechend die Checksumme generiert. Nachdem Sie ein Listing eingegeben haben, sollten Sie es aus Sicherheitsgründen vor dem Starten abspeichern. Sie brauchen hierfür jedoch nicht den Checksummer 64 zu desaktivieren.

Hinweise zum Lesen von Listings

Die Listings haben sich ein wenig im Ausdruckformat verändert, um Ihnen das Eingeben von Programmen wesentlich zu erleichtern.

— Cursorsteuerzeichen und andere Steuerzeichen, die schwer zu lesen sind, werden von nun an in Klartext in speziellen Klammern gesetzt.

Tritt mehrmals hintereinander dasselbe Steuerzeichen auf, so wird diese Steuerzeichen-Sequenz zusammengefaßt, indem zuerst das Steuerzeichen und dann die Anzahl der Wiederholungen dieses Steuerzeichens in Klartext ausgegeben wird.

— alle Commodore-Grafikzeichen, die über Shift zu erreichen sind, werden nicht mehr als Grafikzeichen, sondern als Klartextzeichen dargestellt. Dabei wird aus dem Zeichen, das Sie auf dem Bildschirm sehen, wenn Sie die Tastenkombination Shift und »A« ansprechen, wieder ein »A«. Um dieses »A« vom normalen »A« unterscheiden zu können, ist es etwas kleiner als das gewöhnliche »A« und ist außerdem mit einem Unterstreichungszeichen versehen. Diese Vereinbarung gilt auch für sämtliche andere Commodore-Grafikzeichen, die über Shift zu erreichen sind.

— entsprechendes gilt für sämtliche Commodore-Grafikzeichen, die über die Commodore-Taste zu erreichen sind. Hier wird jedoch das jeweilige Klartextzeichen nicht unterstrichen, sondern überstrichen.

Erläuterungen zu den Cursorsteuerzeichen

Cursorsteuerzeichen werden, wie schon oben erwähnt, umdefiniert. Sie sehen hier eine Liste der möglichen Ausdrücke, die für ein Cursorsteuerzeichen im Listing auftauchen können. Gleichzeitig ersehen Sie aus der Tabelle, welche Taste beziehungsweise Tastenkombination zu drücken ist, damit dieses Steuerzeichen richtig in Ihr Programm übernommen wird. Beachten Sie, daß Sie die Steuercodes nur dann als reverses Zeichen sehen können, wenn der Rechner im »Quote-Modus« arbeitet, das heißt, daß er sich im Gänsefüßchenmodus befindet.

Wenn Sie lesen	! drücken Sie
CTRL steht für Control-Taste, so bedeutet[CTRL-A], daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:	
[down]	! Taste neben rechtem Shift, Cursor unten
[up]	! Shift-Taste & Taste neben rechtem Shift, Cursor hoch
[clear]	! Shift-Taste & 2. Taste ganz rechts oben
[inst]	! Shift-Taste & Taste ganz rechts oben
[home]	! 2. Taste von ganz rechts oben
[del]	! Taste ganz rechts oben
[right]	! Taste ganz rechts unten
[left]	! Shift-Taste & Taste unten rechts
[space]	! Leertaste
[f1]	! grauer Tastenblock rechts
[f3]	! grauer Tastenblock rechts
[f5]	! grauer Tastenblock rechts
[f7]	! grauer Tastenblock rechts
[f2]	! grauer Tastenblock rechts & Shift
[f4]	! grauer Tastenblock rechts & Shift
[f6]	! grauer Tastenblock rechts & Shift
[f8]	! grauer Tastenblock rechts & Shift
[return]	! Shift-Taste & Return
[black]	! Control-Taste & 1
[white]	! Control-Taste & 2
[red]	! Control-Taste & 3
[cyan]	! Control-Taste & 4
[purple]	! Control-Taste & 5
[green]	! Control-Taste & 6
[blue]	! Control-Taste & 7
[yellow]	! Control-Taste & 8
[rvson]	! Control-Taste & 9
[rvoff]	! Control-Taste & 0
[orange]	! Commodore-Taste & 1
[brown]	! Commodore-Taste & 2
[lig.red]	! Commodore-Taste & 3
[grey 1]	! Commodore-Taste & 4
[grey 2]	! Commodore-Taste & 5
[lig.green]	! Commodore-Taste & 6
[lig.blue]	! Commodore-Taste & 7
[grey 3]	! Commodore-Taste & 8

Wenn Sie sich erst einmal an die in Klartext geschriebenen Steuerzeichen gewöhnt haben, werden Sie den Vorteil dieser Schreibweise erkennen. Der zu dem jeweiligen Steuerzeichen gehörende Klartext ist so verfaßt, daß Sie leicht die Taste beziehungsweise die Tastenkombination finden, die Sie drücken müssen.

Checksummer VC 20

Der Checksummer VC 20 ist im Prinzip genauso aufgebaut wie der Checksummer 64. Da beim VC 20 jedoch nicht die Möglichkeit besteht, das ROM softwaremäßig zu modifizieren, mußte ein anderer Weg als beim Commodore 64 gewählt werden, um die Checksumme zu generieren.

In ihrer Funktionsweise unterscheiden sich der Checksummer VC 20 und der Checksummer 64 nicht. Es gelten folgende Sonderregelungen bei der Benutzung des Checksummer VC 20:

— da der Basic-Bereich nicht belegt werden soll, ist das Programm im Kassettenpuffer abgelegt.

— angeschaltet wird der Checksummer VC 20 mit »SYS 955«

— Abschaltung des Checksummer VC 20 wird mit »SYS 58459« vollzogen

ACHTUNG: Nehmen Sie keine Kassetten-Operationen vor, wenn der Checksummer VC 20 eingeschaltet ist. Da das Betriebssystem den Kassettenpuffer mit Daten belegt, kann der Checksummer VC 20 überschrieben werden, was zur Folge hat, daß sich der Rechner bei aktiviertem Checksummer VC 20 »aufhängt«. Wollen Sie deshalb ein Programm auf (von) Kassette abspeichern (laden), so müssen Sie erst den Checksummer VC 20 abschalten (SYS 58459).

Daraufhin kann der Kassettenpuffer mit Daten überschrieben werden, ohne daß der Rechner »aussteigt«.

Als Sicherung wird bei der Initialisierung geprüft, ob das zuletzt angesprochene Peripherie-Gerät der Kassettenrecorder war. Ist das der Fall, so werden die Betriebssystemroutinen LOAD und SAVE für die Benutzung gesperrt. Der Rechner meldet bei Aufruf einer dieser beiden Routinen READY, ohne weitere Aktionen durchzuführen. Diese Sicherung kann man nach

der Tipparbeit aufheben, wenn man den Checksummer VC 20 mit SYS 58459 abschaltet. Dadurch wird der Kassettenpuffer für andere Daten freigemacht. Weiterhin wird dann durch gleichzeitiges Drücken der Tasten »Run-Stop & Restore« erreicht, daß die Betriebssystemroutinen LOAD und SAVE wieder eingerichtet werden.

— Bei Benutzung einer Diskettenstation brauchen Sie nicht darauf zu achten, daß bei LOAD beziehungsweise SAVE der Checksummer VC 20 überschrieben wird, da der Kassettenpuffer für die Diskettenstation normalerweise nicht genutzt wird. Deshalb können Sie die beiden Routinen weiterhin normal nutzen, sofern der Rechner bei der Initialisierung des Checksummer VC 20 feststellt, daß das zuletzt angesprochene Peripherie-Gerät nicht der Kassettenrecorder war.

— bedingt durch den anderen Aufbau des Checksummer VC 20 wird anders als beim Checksummer 64 nach der LOAD-Routine keine Checksumme ausgegeben.

— wird eine Zeile gelöscht, also eine Zahl zwischen 0 und 63999 eingegeben, und danach Return gedrückt, so wird eine Checksumme ausgegeben, die aber keine Bedeutung hat.

Viel Spaß beim Eintippen von Programmen mit dem neuen Checksummer!

(F. Lonczewski / gk)

```
10 REM ********************************     <175>
20 REM *                              *     <247>
30 REM *        CHECKSUMMER 64        *     <162>
40 REM *                              *     <011>
50 REM *            64'ER             *     <061>
60 REM *                              *     <031>
70 REM *         COMMODORE 64         *     <056>
80 REM *                              *     <051>
90 REM ********************************     <255>
100 PRINT"[CLEAR,SPACE13,RVSON]CHECKSUMMER
   [SPACE]64[RVOFF]"                         <025>
110 PRINT                                    <007>
120 PRINT"[DOWN2,SPACE4]ICH[SPACE]ARBEITE!
   [SPACE]BITTE[SPACE]ETWAS[SPACE]GEDULD." <071>
130 FOR I=40960 TO 49151:POKE I,PEEK(I):NEXT
                       <007>
140 FOR I=57344 TO 65535:POKE I,PEEK(I):NEXT
                       <025>
150 POKE 1,53:POKE 42289,96:POKE 42290,228 <001>
160 FOR I=58464 TO 58554:GOSUB 220:POKE I,A
                       <184>
170 PS=PS+A+1:NEXT I                         <109>
180 IF PS<>11187 THEN PRINT"PRUEFSUMMENFEHLER
   [SPACE]!":END                             <180>
190 PRINT"[DOWN4,SPACE9]CHECKSUMMER[SPACE]
   AKTIVIERT."                               <247>
200 PRINT"[DOWN2]AUSSCHALTEN[SPACE]
   :[SPACE]POKE1,55"                         <050>
210 PRINT"[DOWN]ANSCHALTEN[SPACE2]
   :[SPACE]POKE1,53":NEW                     <171>
220 READ A$:IF LEN(A$)<>2 THEN PRINT"TIPPFEHLER
   [SPACE]IN[SPACE]ZEILE"PEEK(63)+PEEK(64)*256
   :END                                      <201>
230 A1=ASC(A$):A2=ASC(RIGHT$(A$,1))          <216>
240 IF A1<48 OR A1>57 THEN IF A1<65 OR A1>70 TH
   EN 310                                    <130>
250 IF A2<48 OR A2>57 THEN IF A2<65 OR A2>70 TH
   EN 310                                    <144>
260 IF A1>64 THEN A1=A1-55:GOTO 280          <204>
270 IF A1<58 THEN A1=A1-48                   <128>
280 IF A2>64 THEN A2=A2-55:GOTO 300          <220>
290 IF A2<58 THEN A2=A2-48                   <151>
300 A=A1*16+A2:RETURN                        <138>
310 PRINT"UNGUELTIGER[SPACE]HEXCODE[SPACE]IN
   [SPACE]ZEILE"PEEK(63)+PEEK(64)*256:END  <021>
320 DATA A0,02,A9,00,85,02,B1,5F             <098>
330 DATA F0,0F,C9,20,D0,03,C8,D0             <146>
340 DATA F5,18,65,02,85,02,4C,6E             <126>
350 DATA E4,C0,04,30,F1,C6,D6,A5             <169>
360 DATA D6,48,A2,03,A9,20,9D,01             <150>
370 DATA 04,BD,B7,E4,20,D2,FF,CA             <238>
380 DATA 10,F2,A6,02,A9,00,20,CD             <169>
390 DATA BD,A9,3E,20,D2,FF,68,85             <245>
400 DATA D6,20,6C,E5,A9,8D,20,D2             <229>
410 DATA FF,4C,80,A4,5C,48,20,C9             <244>
420 DATA FF,AA,68,90,01,8A,60,09             <234>
430 DATA 3C,12,13                            <199>
```

Der Checksummer für den Commodore 64

```
10 REM****************                       <057>
20 REM*              *                       <247>
30 REM* CHECKSUMMER  *                       <056>
40 REM* VERSION VC20 *                       <044>
50 REM*              *                       <021>
60 REM****************                       <107>
70 PRINT"[CLEAR,SPACE2,RVSON]CHECKSUMMER[SPACE2]
   VC-20[RVOFF]"                             <185>
80 PRINT                                     <233>
90 PRINT"[DOWN]EINEN[SPACE]MOMENT,[SPACE]
   BITTE..."                                 <181>
100 FOR I=827 TO 993:GOSUB 180:POKE I,A      <177>
110 PS=PS+A+1:NEXT I                         <049>
120 IF PS<>20612 THEN PRINT"[DOWN]
   PRUEFSUMMENFEHLER[SPACE]!":END            <130>
130 SYS 955:PRINT"CHECKSUMMER[SPACE]AKTIVIERT."
                          <242>
140 PRINT"AN[SPACE]:SYS955"                  <212>
150 PRINT"[DOWN]AUS:SYS58459,[SPACE]BEI[SPACE]
   CAS-[SPACE4]SETTE[SPACE]ZUSAETZLICH[SPACE5]
   RUN/STOP[SPACE]&[SPACE]RESTORE"           <068>
160 PRINT"[DOWN]BEI[SPACE]AKTIVIERTEM[SPACE]
   CHECK-SUMMER[SPACE]KEIN";                 <105>
170 PRINT"[SPACE]CASSETTEN-BETRIEB[SPACE](LOAD,
   [SPACE]SAVE)[SPACE2]ERLAUBT!":NEW         <051>
180 READ A$:IF LEN(A$)<>2 THEN PRINT"TIPPFEHLER
   [SPACE]IN[SPACE]ZEILE"PEEK(63)+PEEK(64)*256
   :END                                      <161>
190 A1=ASC(A$):A2=ASC(RIGHT$(A$,1))          <176>
200 IF A1<48 OR A1>57 THEN IF A1<65 OR A1>70 TH
   EN 270                                    <095>
210 IF A2<48 OR A2>57 THEN IF A2<65 OR A2>70 TH
   EN 270                                    <109>
220 IF A1>64 THEN A1=A1-55:GOTO 240          <159>
230 IF A1<58 THEN A1=A1-48                   <087>
240 IF A2>64 THEN A2=A2-55:GOTO 260          <184>
250 IF A2<58 THEN A2=A2-48                   <110>
260 A=A1*16+A2:RETURN                        <098>
270 PRINT"UNGUELTIGER[SPACE]HEXCODE[SPACE]IN
   [SPACE]ZEILE"PEEK(63)+PEEK(64)*256:END  <237>
280 DATA 20,5F,03,86,7A,84,7B,20             <061>
290 DATA 73,00,AA,F0,F3,A2,FF,86             <130>
300 DATA 3A,90,0A,A2,00,86,FF,20             <097>
310 DATA 79,C5,4C,E1,C7,A2,01,86             <127>
320 DATA FF,4C,9C,C4,A6,FF,E0,01             <199>
330 DATA F0,03,4C,60,C5,A0,02,A9             <125>
340 DATA 00,85,FE,B1,5F,F0,0F,C9             <186>
350 DATA 20,D0,03,C8,D0,F5,18,65             <141>
360 DATA FE,85,FE,4C,76,03,C0,04             <193>
370 DATA 30,F1,C6,D6,A5,D6,48,A2             <198>
380 DATA 03,A9,20,9D,01,04,BD,B7             <180>
390 DATA 03,20,D2,FF,CA,10,F2,A6             <217>
400 DATA FE,A9,00,20,CD,DD,A9,3E             <016>
410 DATA 20,D2,FF,68,85,D6,20,87             <220>
420 DATA E5,A9,8D,20,D2,FF,A2,00             <003>
430 DATA 86,FF,F0,AE,09,3C,12,13             <002>
440 DATA A9,3B,8D,02,03,A9,03,8D             <249>
450 DATA 03,03,A5,BA,C9,01,D0,10             <235>
460 DATA A9,74,8D,30,03,8D,32,03             <239>
470 DATA A9,C4,8D,31,03,8D,33,03             <007>
480 DATA AD,88,02,8D,90,03,60                <113>
```

Der Checksummer für den VC 20

MSE — Abtippen sicher und leicht gemacht

Ähnlich wie der »Checksummer« ist auch der MSE ein Hilfsmittel bei der Eingabe von Listings, diesmal jedoch bei reinen Maschinensprache-Programmen.

Im Gegensatz zum »Checksummer« aber ist die Eingabe nicht ohne den MSE möglich. Der MSE verringert die Tipparbeit um ein Drittel und schließt Fehleingaben vollkommen aus. Außerdem können Sie die DATAs blind eingeben, ohne andauernd auf den Bildschirm schauen zu müssen. Dies wird durch akustische Meldungen realisiert.

Sicher kennen Sie die Situation: Man hat ein langes Listing mit DATA-Zeilen abgetippt, versucht es, das erste Mal zu starten und — nichts läuft. Dann beginnt nach der mühseligen Tipperei die noch mühseligere Fehlersuche. Als letzter Ausweg bleibt dann nur noch der Anruf bei der Redaktion oder dem Verfasser, ob vielleicht doch ein Druckfehler...

Damit ist es jetzt vorbei. Ab dieser Ausgabe werden die Maschinenprogramme im 64'er mit dem MSE abgedruckt. MSE ist ein Maschinenspracheditor, mit dem ein Vertippen ausgeschlossen ist. Eine abgetippte Zeile wird nur angenommen, wenn sie richtig ist. Wie ist das möglich ? Eine Checksumme am Ende jeder Zeile prüft, ob die richtigen Werte in der richtigen Zeile an der richtigen Stelle stehen. Wenn nicht, ertönt ein Warnsignal, und man beseitigt den Fehler.

War die Zeile korrekt, erklingt ein Gong, und die nächste Zeilennummer wird ausgegeben. Damit ist also auch »blindes« Eintippen möglich; Sie können sich voll auf den Text konzentrieren.

MSE verringert die Tipparbeit um ein Drittel. Anstelle von dreistelligen DATAs brauchen Sie nur noch zweistellige Hex-Zahlen einzugeben, die direkt in den Speicher gePOKEt werden.

So arbeitet man mit MSE

Laden und starten Sie MSE. Zuerst wird der Programmname und die Start- und Endadresse erfragt. Diese Angaben entnehmen Sie dem Kopf des jeweiligen abgedruckten Listings. MSE meldet sich dann mit der Zeilennummer der ersten Zeile. Wenn Sie die Zeile richtig eingegeben haben, erscheint die nächste Zeilennummer und so weiter bis zum Ende. Zum Schluß wird das fertige Programm mit »CTRL-S« auf Diskette oder Kassette abgespeichert. Dazu sind keine weiteren Angaben mehr erforderlich. Das Programm kann dann ganz normal wieder absolut geladen und gestartet werden. Wenn Sie nicht alles auf einmal tippen wollen, können Sie jederzeit unterbrechen und den eingetippten Teil mit »CTRL-S« abspeichern. Wollen Sie weiterarbeiten, laden und starten Sie MSE wieder. Geben Sie auf die Frage nach der Startadresse aber jetzt »L« ein, um Ihr Teilprogramm zu laden. Jetzt können Sie mit »CTRL-N« die Adresse eingeben, an der Sie weitertippen müssen. Wenn Sie sich nicht gemerkt haben, wie weit Sie gekommen sind, geben Sie nach dem Laden »CTRL-M« ein.

Auf die Frage nach der Startadresse antworten Sie mit der Anfangsadresse, die links in der Kopfzeile auf dem Bildschirm steht. Nun wird Ihr Programm aufgelistet. Mit »SPACE« wird das Listen fortgesetzt, mit »STOP« abgebrochen. Das Ende Ihres Programmteils erkennen Sie sehr einfach daran, daß nur noch der Wert »AA« in der Zeile steht. Die Adresse dieser Zeile müssen Sie anschließend mit »CTRL-N« eingeben. Das Programm ist nur mit »STOP/RESTORE« zu verlassen. Speichern Sie aber vorher unbedingt immer Ihren Text ab.

Wollen Sie selbst Programme mit MSE ausdrucken, laden Sie Ihr Programm wie oben beschrieben und geben anschließend »CTRL-P« ein. Die Druckausgabe läßt sich mit »STOP« jederzeit abbrechen.

Hinweise zum Abtippen

Vor dem Abtippen oder späteren Wiederladen des MSE-Laders müssen Sie unbedingt folgende Zeile eingeben:
POKE 43,1: POKE 44,32: POKE 8192,0: NEW

Beachten Sie bei der Eingabe die Hinweise im »Checksummer«. Speichern Sie den »MSE Lader« nach dem Abtippen unbedingt ab. Starten Sie das Programm mit RUN. Fehlerhafte Zeilen werden angezeigt und müssen korrigiert werden, bis der Lader zum »READY« durchläuft. Jetzt müssen Sie das fertige MSE-Programm abspeichern. Dazu brauchen Sie nur »RETURN« zu drücken, weil die erforderlichen Angaben schon auf dem Bildschirm stehen. (Kassettenbesitzer müssen in Zeile 343 die letzte Zahl in »1« abändern). Ab jetzt können Sie »MSE V1.0« direkt, also ohne den DATA-Lader benutzen. MSE V1.0 wird ganz normal mit »,8« geladen. Heben Sie das Programm gut auf, Sie werden es noch häufig brauchen.

(N. Mann/D. Weineck/gk)

MSE-Befehle:

DEL	löscht die letzte Eingabe.
CTRL-S	speichert das eingetippte Programm ab.
CTRL-L	lädt ein Programm. Start- und Endadresse werden automatisch ermittelt.
CTRL-M	listet den Speicherinhalt. Abbruch mit STOP-Taste, weiter mit Leertaste
CTRL-N	erlaubt die Eingabe einer neuen Adresse zum Weitertippen.
CTRL-P	gibt ein MSE-Listing auf dem Drucker aus.

Die Befehle des MSE auf einen Blick

Listing MSE. Dieses Programm erleichtert Ihnen die Eingabe von Maschinenprogrammen ganz erheblich. Sie sparen Zeit und machen keine Fehler mehr.

```
1 REM ********************************   <208>
2 REM *     ++++ MSE - LADER   ++++   *   <183>
3 REM *              VON              *   <217>
4 REM *  D.WEINECK & N.MANN           *   <043>
5 REM *  FLEETRADE 40, 2800 BREMEN 1  *   <184>
6 REM *  TEL. 0421/493090/231401      *   <133>
7 REM ********************************   <214>
8 :                                        <066>
9 :                                        <067>
10 DIM H(75) : FOR I=0 TO 9                <088>
20 H(48+I)=I : H(65+I)=I+10 : NEXT         <250>
30 FOR I=2048 TO 3755 : READ A$            <006>
40 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))  <063>
50 D=H(H)*16+H(L) : S=S+D : POKE I,D       <181>
60 A=A+1:IF A<9  THEN NEXT : A=-1          <111>
65 PRINT "ZEILE:";1000+Z;                  <012>
70 READ V : Z=Z+1 : IF V=S THEN 85         <210>
80 PRINT" PRUEFSUMMENFEHLER !";999+Z:STOP  <015>
85 IF A<0 THEN END                         <043>
90 S=0 : A=0 : PRINT : NEXT : END          <053>
95 :                                       <153>
96 :                                       <154>
1000 DATA 00,0B,08,0A,00,9E,32,30,36, 339  <083>
1001 DATA 31,00,00,00,A2,08,A9,36,85, 575  <075>
1002 DATA A4,A9,08,85,A5,A9,00,85,A6, 1107 <189>
1003 DATA A9,B0,85,A7,A0,00,B1,A4,91, 1291 <190>
1004 DATA A6,C8,D0,F9,E6,A5,E6,A7,CA, 1817 <028>
1005 DATA D0,F2,A9,36,85,01,4C,00,B0, 1059 <180>
1006 DATA 20,D1,B1,A9,06,8D,21,D0,A9, 1144 <193>
1007 DATA 03,8D,20,D0,8D,86,02,A0,B3, 1000 <169>
1008 DATA A9,74,20,FF,B1,A0,B3,A9,B9, 1442 <238>
1009 DATA 20,FF,B1,A0,00,20,CF,FF,99, 1271 <233>
1010 DATA 01,02,C8,C9,0D,D0,F5,88,F0, 1246 <212>
1011 DATA D2,C0,0F,90,02,A0,0E,8C,00, 877  <150>
1012 DATA 02,20,EA,B1,A0,B3,A9,CF,20, 1192 <219>
1013 DATA FF,B1,20,8E,B4,85,FC,85,62, 1402 <237>
1014 DATA 20,8E,B4,85,FB,85,61,20,A7, 1167 <207>
1015 DATA B4,D0,20,A0,B3,A9,E5,20,FF, 1444 <234>
1016 DATA B1,20,8E,B4,85,60,20,8E,B4, 1114 <193>
1017 DATA 85,5F,20,A7,B4,D0,0A,A5,61, 1087 <213>
1018 DATA C5,5F,A5,62,E5,60,90,06,20, 1062 <183>
1019 DATA 43,B3,4C,3A,B0,A9,AA,A0,00, 1055 <222>
1020 DATA 91,FB,E6,FB,D0,02,E6,FC,20, 1601 <007>
1021 DATA 3F,B2,90,EF,4C,FB,B4,A2,02, 1295 <011>
1022 DATA 86,58,A9,A6,A0,9D,20,F2,B1, 1325 <226>
1023 DATA 20,E4,FF,F0,FB,C9,30,90,0C, 1411 <248>
1024 DATA C9,47,B0,08,C9,3A,90,0B,C9, 1071 <228>
1025 DATA 41,B0,07,C9,14,D0,0F,4C,0B, 779  <173>
1026 DATA B1,20,D2,FF,A6,58,95,F7,C6, 1522 <254>
1027 DATA 58,D0,D2,60,AE,8D,02,F0,26, 1197 <231>
1028 DATA C9,0C,D0,03,4C,0B,B6,C9,13, 913  <187>
1029 DATA D0,03,4C,8B,B5,C9,0D,D0,03, 1032 <228>
1030 DATA 4C,BA,B4,C9,10,D0,03,4C,68, 1050 <232>
1031 DATA B5,C9,0E,D0,06,20,5F,B4,4C, 993  <203>
1032 DATA 64,B1,4C,92,B0,A5,F9,20,02, 1123 <204>
1033 DATA B1,0A,0A,0A,0A,85,F9,A5,FB, 1012 <247>
1034 DATA 20,02,B1,05,F9,60,C9,3A,90, 964  <154>
1035 DATA 02,69,08,29,0F,60,A6,59,E0, 746  <153>
1036 DATA 08,90,1F,A6,58,E0,02,B0,06, 845  <155>
1037 DATA 20,D2,FF,4C,8E,B0,C6,59,A0, 1338 <017>
1038 DATA 14,A9,92,20,F2,B1,CA,D0,FA, 1446 <006>
1039 DATA 84,57,68,68,4C,8B,B1,A6,D3, 1196 <249>
1040 DATA E0,08,B0,03,4C,92,B0,20,D2, 1051 <200>
1041 DATA FF,A6,58,E0,02,90,09,C6,59, 1175 <242>
1042 DATA 20,D2,FF,C6,58,D0,F9,4C,8E, 1458 <040>
1043 DATA B0,48,4A,4A,4A,4A,20,59,B1, 842  <185>
1044 DATA 68,29,0F,C9,0A,90,02,69,06, 628  <163>
1045 DATA 69,30,4C,D2,FF,A2,FC,9A,20, 1294 <027>
1046 DATA D1,B1,20,48,B2,20,EA,B1,20, 1143 <217>
1047 DATA 9F,B2,A5,FC,20,4E,B1,A5,FB, 1457 <057>
1048 DATA 20,4E,B1,20,ED,B1,A9,3A,A0, 1120 <250>
1049 DATA 20,20,F2,B1,A9,00,85,59,20, 906  <145>
1050 DATA 8E,B0,20,ED,B1,A4,59,20,EF, 1288 <029>
1051 DATA B0,91,FB,C8,84,59,C0,08,90, 1337 <249>
1052 DATA EC,20,10,B2,A9,12,20,D2,FF, 1146 <251>
1053 DATA 20,8E,B0,20,EF,B0,C5,FF,F0, 1489 <048>
1054 DATA 0D,20,43,B3,A9,14,A0,14,20, 692  <155>
1055 DATA F2,B1,4C,A2,B1,A9,92,20,D2, 1391 <005>
1056 DATA FF,20,33,B2,20,E0,B2,20,3F, 1045 <203>
1057 DATA B2,90,9F,4C,8B,B5,A9,93,20, 1225 <010>
1058 DATA D2,FF,A2,00,A9,03,9D,00,D8, 1172 <011>
1059 DATA 9D,00,D9,9D,00,DA,9D,00,DB, 1125 <030>
1060 DATA E8,D0,EF,60,A9,0D,2C,A9,20, 1202 <029>
1061 DATA 4C,D2,FF,20,D2,FF,98,4C,D2, 1476 <069>
1062 DATA FF,20,E4,FF,F0,FB,60,84,5D, 1582 <069>
1063 DATA 85,5C,A0,00,B1,5C,F0,06,20, 932  <183>
1064 DATA D2,FF,C8,D0,F6,60,A5,FB,85, 1764 <075>
1065 DATA 5A,A0,00,84,5B,B1,FB,18,65, 1026 <254>
1066 DATA 5A,85,5A,90,02,E6,5B,06,5A, 876  <212>
1067 DATA 26,5B,C8,C0,08,90,EC,A5,5A, 1164 <028>
1068 DATA 65,5B,85,FF,60,18,A5,FB,69, 1221 <028>
1069 DATA 08,85,FB,90,02,E6,FC,60,A5, 1281 <020>
1070 DATA FB,C5,5F,A5,FC,E5,60,60,A0, 1541 <061>
1071 DATA B3,A9,FB,20,FF,B1,A0,01,B9, 1409 <053>
1072 DATA 00,02,20,D2,FF,CC,00,02,C8, 905  <202>
1073 DATA 90,F4,A9,10,ED,00,02,AA,20, 1014 <247>
1074 DATA ED,B1,CA,D0,FA,A5,62,20,4E, 1447 <073>
1075 DATA B1,A5,61,20,4E,B1,20,ED,B1, 1172 <013>
1076 DATA A5,60,20,4E,B1,A5,5F,20,4E, 918  <223>
1077 DATA B1,A9,9F,20,D2,FF,20,EA,B1, 1445 <065>
1078 DATA 24,5E,10,01,60,A9,12,20,D2, 672  <165>
1079 DATA FF,A2,28,20,ED,B1,CA,D0,FA, 1563 <095>
1080 DATA A9,92,4C,D2,FF,A5,D6,C9,16, 1458 <078>
1081 DATA B0,01,60,A9,A0,85,A4,A9,78, 1188 <013>
1082 DATA 85,A6,A9,04,85,A5,85,A7,A2, 1232 <018>
1083 DATA 13,A0,27,B1,A4,91,A6,88,10, 1022 <235>
1084 DATA F9,CA,F0,19,18,A5,A4,69,28, 1214 <039>
1085 DATA 85,A4,90,02,E6,A5,18,A5,A6, 1193 <018>
1086 DATA 69,28,85,A6,90,E0,E6,A7,4C, 1285 <038>
1087 DATA B6,B2,A9,91,4C,D2,FF,A9,0F, 1399 <097>
1088 DATA 8D,18,D4,A9,00,8D,05,D4,A9, 1073 <040>
1089 DATA F7,8D,06,D4,A9,11,8D,04,D4, 1149 <046>
1090 DATA A9,32,8D,01,D4,A9,00,8D,00, 883  <226>
1091 DATA D4,A0,B0,20,09,B3,A9,10,8D, 1046 <009>
1092 DATA 04,D4,60,A2,FF,CA,D0,FD,88, 1528 <090>
1093 DATA D0,F8,60,A9,0F,8D,18,D4,A9, 1282 <068>
1094 DATA 2D,8D,05,D4,A9,A5,8D,06,D4, 1096 <066>
1095 DATA A9,21,8D,04,D4,A9,07,8D,01, 877  <243>
1096 DATA D4,A9,05,8D,00,D4,A0,FF,20, 1186 <053>
1097 DATA 09,B3,A9,20,8D,04,D4,A9,00, 915  <231>
1098 DATA 8D,01,D4,8D,00,D4,60,38,20, 891  <219>
1099 DATA F0,FF,8A,48,98,48,18,A0,06, 1119 <046>
1100 DATA A2,18,20,F0,FF,A0,B4,A9,0A, 1232 <057>
1101 DATA 20,FF,B1,20,12,B3,20,E4,FF, 1208 <045>
1102 DATA F0,FB,A2,1D,A9,14,20,D2,FF, 1368 <092>
1103 DATA CA,D0,FA,68,A8,68,AA,18,4C, 1306 <098>
1104 DATA F0,FF,0D,0D,0D,20,20,20,20, 662  <231>
1105 DATA 20,20,20,4D,41,53,43,48,49, 533  <170>
1106 DATA 4E,45,4E,53,50,52,41,43,48, 674  <205>
1107 DATA 45,20,2D,20,45,44,49,54,4F, 551  <197>
1108 DATA 52,20,0D,0D,20,20,20,20,20, 300  <149>
1109 DATA 20,20,20,56,4F,4E,20,4E,2E, 495  <224>
1110 DATA 4D,41,4E,4E,20,26,20,44,2E, 514  <221>
1111 DATA 57,45,49,4E,45,43,4B,00,0D, 531  <216>
1112 DATA 0D,0D,20,20,20,50,52,4F,47, 434  <197>
1113 DATA 52,41,4D,4D,4E,41,4D,45,20, 622  <227>
1114 DATA 3A,20,00,0D,0D,20,20,20,53, 295  <185>
1115 DATA 54,41,52,54,41,44,52,45,53, 682  <177>
1116 DATA 53,45,20,3A,20,24,00,0D,0D, 336  <194>
1117 DATA 20,20,20,45,4E,44,41,44,52, 526  <177>
1118 DATA 45,53,53,45,20,20,20,3A,20, 490  <172>
1119 DATA 24,00,92,05,20,50,52,4F,47, 531  <180>
1120 DATA 52,41,4D,4D,20,3A,20,00,12, 441  <195>
1121 DATA 20,20,2A,2A,2A,20,46,41,4C, 433  <211>
1122 DATA 53,43,48,45,20,45,49,4E,47, 614  <208>
1123 DATA 41,42,45,20,2A,2A,2A,20,20, 422  <193>
1124 DATA 92,00,0D,0D,2A,2A,2A,20,45, 399  <243>
1125 DATA 4E,44,45,20,2A,2A,2A,00,13, 392  <223>
1126 DATA 05,20,20,12,44,92,49,53,4B, 532  <189>
1127 DATA 20,4F,44,45,52,20,12,54,92, 610  <190>
1128 DATA 41,50,45,0D,00,13,20,20,49, 383  <181>
1129 DATA 2F,4F,20,2D,20,46,45,48,4C, 522  <247>
```

Listing von MSE (Schluß)

```
1130 DATA 45,52,00,20,D1,B1,20,48,B2, 851  <215>
1131 DATA A0,B3,A9,CF,20,FF,B1,20,8E, 1353 <115>
1132 DATA B4,85,FC,20,8E,B4,85,FB,C5, 1500 <115>
1133 DATA 61,A5,FC,E5,62,90,23,A5,FB, 1436 <098>
1134 DATA C5,5F,A5,FC,E5,60,B0,19,20, 1267 <097>
1135 DATA A7,B4,D0,14,60,20,A7,B4,F0, 1290 <065>
1136 DATA 0C,85,F9,20,A7,B4,F0,05,85, 1151 <066>
1137 DATA F8,4C,EF,B0,68,68,20,43,B3, 1225 <090>
1138 DATA 4C,5F,B4,20,CF,FF,C9,4C,D0, 1330 <146>
1139 DATA 09,20,D1,B1,20,48,B2,4C,0B, 796  <010>
1140 DATA B6,C9,0D,60,A9,00,85,5E,20, 920  <019>
1141 DATA 5F,B4,20,EA,B1,20,0D,B5,24, 980  <040>
1142 DATA 5E,30,05,20,E4,FF,F0,FB,20, 1185 <097>
1143 DATA E1,FF,F0,26,20,9F,B2,24,5E, 1257 <110>
1144 DATA 10,09,20,4E,B5,20,0D,B5,20, 574  <246>
1145 DATA 60,B5,20,33,B2,20,3F,B2,90, 955  <000>
1146 DATA D7,A0,B4,A9,28,20,FF,B1,20, 1260 <095>
1147 DATA E4,FF,C9,0D,D0,F9,A9,00,85, 1456 <141>
1148 DATA 5E,A5,61,85,FB,A5,62,85,FC, 1388 <131>
1149 DATA 20,E0,B2,4C,64,B1,A5,FC,20, 1236 <092>
1150 DATA 4E,B1,A5,FB,85,FF,20,4E,B1, 1346 <144>
1151 DATA A9,20,A0,3A,20,F2,B1,A0,00, 1030 <053>
1152 DATA 20,ED,B1,B1,FB,20,4E,B1,C8, 1361 <128>
1153 DATA C0,08,90,F3,20,ED,B1,24,5E, 1163 <090>
1154 DATA 30,03,A9,12,2C,A9,20,20,D2, 725  <255>
1155 DATA FF,20,10,B2,A5,FF,20,4E,B1, 1188 <123>
1156 DATA A9,92,20,D2,FF,4C,EA,B1,A9, 1468 <157>
1157 DATA FF,85,B8,85,B9,A9,04,85,BA, 1382 <143>
1158 DATA 20,C0,FF,A2,FF,4C,C9,FF,20, 1460 <165>
1159 DATA CC,FF,A9,FF,4C,C3,FF,20,5F, 1536 <215>
1160 DATA B4,A9,80,85,5E,20,4E,B5,20, 1027 <088>
1161 DATA 48,B2,A2,24,A9,2D,20,D2,FF, 1159 <121>
1162 DATA CA,D0,FA,20,EA,B1,20,EA,B1, 1546 <162>
1163 DATA 20,60,B5,4C,C1,B4,20,B8,B5, 1155 <093>
1164 DATA A6,5F,A4,60,A9,61,20,D8,FF, 1290 <131>
1165 DATA B0,0A,20,B7,FF,29,BF,D0,03, 1099 <135>
1166 DATA 4C,FB,B4,A9,01,20,C3,FF,20, 1191 <131>
1167 DATA 68,B6,A0,B4,A9,4F,20,FF,B1, 1338 <147>
1168 DATA 20,F9,B1,4C,FB,B4,20,68,B6, 1283 <130>
1169 DATA A9,37,A0,B4,20,FF,B1,20,F9, 1309 <126>
1170 DATA B1,A2,08,C9,44,F0,06,A2,01, 1025 <079>
1171 DATA C9,54,D0,F1,A9,01,A8,20,BA, 1290 <123>
1172 DATA FF,A0,00,E0,01,F0,1A,A9,40, 1139 <110>
1173 DATA 8D,20,02,A9,3A,8D,21,02,B9, 763  <050>
1174 DATA 01,02,99,22,02,C8,CC,00,02, 598  <013>
1175 DATA 90,F4,C8,CB,D0,0C,B9,01,02, 1196 <121>
1176 DATA 99,20,02,C8,CC,00,02,D0,F4, 1045 <092>
1177 DATA 98,A2,20,A0,02,4C,BD,FF,20, 1060 <119>
1178 DATA 88,B5,A5,BA,C9,08,90,33,A6, 1286 <146>
1179 DATA B9,B6,57,A9,01,20,C3,FF,A9, 1227 <136>
1180 DATA 60,85,B9,20,C0,FF,B0,28,A5, 1274 <126>
1181 DATA BA,20,B4,FF,A5,B9,20,96,FF, 1440 <174>
1182 DATA 20,A5,FF,85,61,A5,90,4A,4A, 1139 <128>
1183 DATA B0,13,20,A5,FF,85,62,20,AB, 1081 <112>
1184 DATA FF,A5,57,85,B9,A9,00,20,D5, 1239 <141>
1185 DATA FF,90,03,4C,A3,B5,86,5F,84, 1183 <144>
1186 DATA 60,A5,BA,C9,01,D0,0A,AD,3D, 1101 <149>
1187 DATA 03,85,61,AD,3E,03,85,62,4C, 778  <063>
1188 DATA FB,B4,A9,13,20,D2,FF,A2,1C, 1306 <168>
1189 DATA 20,ED,B1,CA,D0,FA,60, 1202      <104>
```

VC 20 steuert Super 8-Kamera

Ein kleines Programm und eine schnell aufgebaute Schaltung für alle Trickfilmer.

Beim Erstellen von Trickszenen mit der Super 8-Kamera fallen regelmäßig lästige Rechnungen an: Die Anzahl der bereits aufgenommenen Bilder muß festgehalten, die resultierende effektive Wiedergabezeit muß berechnet werden. Schließlich ist auch das Einhalten einer bestimmten Bildfrequenz bei Einzelbildaufnahmen von Hand kaum möglich.

Da fast jede Super 8-Kamera über einen Anschluß für einen elektrischen Fernauslöser verfügt, bietet sich eine Steuerung per Computer geradezu an. Der Hardware-Aufwand beschränkt sich dabei auf einen Transistor BC-238 (oder ähnlich) als Treiberstufe und ein handelsübliches 5-Volt-Relais, wie es jeder Elektronikladen vorrätig hat. Die beiden Bauteile werden an den User-Port des VC 20 geschaltet. Über die Adresse 37136, dem Ausgaberegister von Port B des User-Port-VIA, läßt sich das externe Relais dann ein- und ausschalten. Die Pin-Belegung des User-Ports kann im Handbuch nachgeschlagen werden.

Der Aufbau der Schaltung ist völlig unkritisch, sollte aber dennoch mit großer Sorgfalt erfolgen, da eine falsche Beschaltung zur Zerstörung des VIA führen kann.

Das Programm zur Kamerasteuerung ist bewußt einfach gehalten und erklärt sich weitgehend von selbst. Die Bildfrequenz kann über die Funktionstasten sehr einfach und schnell gewählt werden. Das Programm informiert ständig über die Gesamtzahl der aufgenommenen Einzelbilder, über die effektive Laufzeit und die gewählte Bildfrequenz.

(Wolf-D. Robrahn/ev)

Die verwendeten Variablen

A	= gewählte Bildfrequenz
C	= Summe der gemachten Bilder
K(X)	= festgelegte Frequenz
P1	= Variable für F-Tasten-Änderung
Q	= Allzweck-Variable
T	= Allzweck-Variable
T2	= Schleifenlänge für An und Aus
X	= momentane Bildanzahl

Anschluß der Schaltung an den User-Port

Listing »Super 8-Steuerung«

```
1 REM SUPER 8 STEUERUNG                            <025>
2 REM                                              <145>
3 REM                                              <146>
4 REM WOLF-D.ROBRAHN                               <118>
5 REM BEETHOVENSTR.18                              <196>
6 REM 2200 ELMSHORN                                <193>
7 REM TEL: 04121/74735                             <230>
8 REM                                              <151>
9 REM                                              <152>
19 POKE 37138,255                                  <120>
20 POKE 37136,0                                    <011>
25 C=0:T2=200                                      <066>
27 REM *********************                       <070>
28 REM BELEGUNG DER F-TASTEN*                      <059>
29 REM *********************                       <072>
30 PRINT"{CLR,4DOWN}BILDFREQUENZ"                   <093>
35 PRINT"{DOWN}FUER{2SPACE}F1-";                   <034>
40 INPUT K(1)                                      <122>
45 PRINT"FUER{2SPACE}F3-";                         <029>
50 INPUT K(2)                                      <133>
55 PRINT"FUER{2SPACE}F5-";                         <041>
60 INPUT K(3)                                      <144>
70 PRINT"FUER{2SPACE}F7-";                         <058>
75 INPUT K(4)                                      <160>
80 REM ************************                    <249>
82 REM AUFBAU DES BILDSCHIRMES *                   <020>
83 REM ************************                    <252>
84 PRINT"{CLR}"                                    <196>
85 PRINT"{HOME,3RIGHT}F1="K(1)" BILDER"            <019>
90 PRINT"{3RIGHT}F3="K(2)" BILDER"                 <008>
95 PRINT"{3RIGHT}F5="K(3)" BILDER"                 <016>
96 PRINT"{3RIGHT}F7="K(4)" BILDER"                 <020>
97 FOR Q=1 TO 22:PRINT"Z";:NEXT                    <006>
100 PRINT"{DOWN,RIGHT}JETZIGE BILDFREQUENZ:"
                        <070>
110 PRINT"{2SPACE}****{2SPACE}"                    <243>
115 PRINT"{DOWN,RIGHT}ZAEHLER:"                    <195>
120 PRINT"{2DOWN,2RIGHT}DRUECKEN SIE EINE
   {9SPACE}F- TASTE{13SPACE}FUER START"            <184>
122 PRINT"{3DOWN}"TAB(7)"BILDER TOTAL"             <007>
123 PRINT"{DOWN}"TAB(7)"LAUFZEIT SEC."             <029>
124 REM *****************                          <255>
125 REM TASTATUR ABFRAGE *                         <150>
126 REM *****************                          <001>
127 GET A$                                         <133>
130 IF A$<>CHR$(133)AND A$<>CHR$(134)AND A$<>CH
   R$(135)AND A$<>CHR$(136)AND A$<>"A"THEN 125
                        <060>
140 IF A$=CHR$(133)THEN A=K(1):GOSUB 1000          <204>
150 IF A$=CHR$(134)THEN A=K(2):GOSUB 1000          <216>
160 IF A$=CHR$(135)THEN A=K(3):GOSUB 1000          <228>
170 IF A$=CHR$(136)THEN A=K(4):GOSUB 1000          <240>
180 IF A$="A"THEN 2000                             <068>
190 REM *****************                          <065>
192 REM STEUERUNGSABLAUF *                         <057>
193 REM *****************                          <068>
200 FOR X=1 TO A                                   <105>
210 POKE 37136,1                                   <202>
212 PRINT"{HOME,10DOWN,10RIGHT,5SPACE}"            <144>
214 PRINT"{HOME,10DOWN,10RIGHT}"X                  <234>
220 POKE 36878,10                                  <016>
230 POKE 36874,231                                 <075>
240 FOR T=1 TO 10:NEXT                             <105>
250 POKE 36874,0:POKE 36878,0                      <054>
260 FOR T=1 TO T2:NEXT                             <163>
270 POKE 37136,0                                   <006>
275 FOR T=1 TO T2:NEXT                             <178>
280 NEXT                                           <155>
285 C=C+A:PRINT"{HOME,19DOWN}"C                    <241>
286 PRINT"{DOWN,6SPACE}"                           <013>
287 PRINT"{UP}"INT(C*10/18)/10                     <008>
290 GOTO 85                                        <025>
997 REM ***************************************
                        <007>
998 REM ZEILEN LOESCHEN + ANZEIGE DER FREQUENZ *
                        <051>
999 REM ***************************************
                        <009>
1000 PRINT"{HOME,8DOWN,2RIGHT,4SPACE}"             <157>
1001 PRINT"{HOME,8DOWN,2RIGHT}"A                   <223>
1002 PRINT"{HOME,13DOWN,62SPACE}"                  <186>
1003 RETURN                                        <124>
1997 REM *************                             <133>
1998 REM AENDERUNGEN *                             <186>
1999 REM *************                             <135>
2000 PRINT"{CLR}SIE WOLLEN DIE BILD-{2SPACE}
   FREQUENZ AENDERN?"                              <191>
2005 GET A$:IF A$<>"J"AND A$<>"N"THEN 2005         <118>
2006 IF A$="N"THEN 3000                            <123>
2010 PRINT"{2DOWN}FUER"                            <018>
2020 PRINT"{8SPACE,RVSON}<1>{RVOFF}= F1"           <203>
2030 PRINT"{DOWN,8SPACE,RVSON}<2>{RVOFF}= F3"
                        <233>
2040 PRINT"{DOWN,8SPACE,RVSON}<3>{RVOFF}= F5"
                        <246>
2050 PRINT"{DOWN,8SPACE,RVSON}<4>{RVOFF}= F7"
                        <004>
2060 INPUT"{2DOWN}";P1                             <187>
2065 PRINT"{UP}ALTE FREQUENZ:"K(P1)                <116>
2070 INPUT"{2DOWN}NEU: ";K(P1)                     <131>
2090 GOTO 80                                       <035>
3000 PRINT"{CLR,2DOWN}SIE WOLLEN DIE BILD-
   {2SPACE}FOLGE AENDERN!"                         <172>
3010 PRINT"{2DOWN}MOMENTANE SCHLEIFEN-{2SPACE}
   LAENGE"T2                                       <096>
3030 INPUT"{2DOWN}NEUE SCHLEIFENLAENGE";T2         <247>
3050 GOTO 80                                       <230>
```

C 16/Plus 4 — Listings gesucht

Die neuen Commodore-Computer C 16/116 und Plus 4 sind inzwischen schon in größeren Stückzahlen verkauft worden. Gehören Sie vielleicht zu den Käufern der ersten Stunde und haben deshalb schon Erfahrungen damit sammeln können? Oder haben Sie vielleicht schon interessante Programme für diese Computer entwickelt?

Dann sollten Sie einmal zur Schreibmaschine (beziehungsweise zur Textverarbeitung) greifen und eine möglichst ausführliche Programmbeschreibung erstellen. Anschließend werfen Sie den Drucker an, um ein Programmlisting zu erzeugen und kopieren dann das Programm auf einen Datenträger (Diskette oder Kassette).

Zum Schluß entwerfen Sie noch ein kurzes Anschreiben, auf dem Sie bitte unbedingt Name, Anschrift und Telefonnummer vermerken und (ganz wichtig!) auch erwähnen, für welchen Computer das Programm geschrieben wurde und welche Erweiterungen und zusätzliche Geräte notwendig sind.

Auf den Briefumschlag, in den Sie diese Unterlagen stecken, schreiben Sie bitte neben Ihrem Absender noch folgenden Text:

An die
Redaktion 64'er
Markt & Technik Verlag AG
Hans-Pinsel-Straße 2
8013 Haar bei München

Ein oder zwei Tage, nachdem Sie den Brief zur Post gebracht haben, kommt er bei uns in der Redaktion an. Wir freuen uns über die Zusendung, testen das Programm und entscheiden über die Veröffentlichung.

So ist allen geholfen: Unsere Leser finden ständig interessante Programme in ihrem 64'er Magazin, wir Redakteure haben Arbeit und Sie bekommen bares Geld — 100, 200, 300 oder auch 2000 Mark.

Ohne gutes Werkzeug geht es nicht: SMON (Teil 4)

Der Trace-Modus, also das Abarbeiten von Maschinenprogrammen Schritt für Schritt, ist das wohl wichtigste Hilfsmittel beim Austesten von Maschinenprogrammen. SMON bietet sogar drei verschiedene Trace-Möglichkeiten, die in diesem vierten Teil dargestellt werden. Außerdem erhalten Sie eine Übersicht aller Befehle, und wir zeigen Ihnen, wie Sie SMON in einen anderen Bereich verschieben können.

Einen Befehl haben wir Ihnen bisher unterschlagen, der zwar bereits vorhanden, aber noch nicht beschrieben war. Es handelt sich um den Vergleich zweier Speicherbereiche. Die Syntax ist sehr einfach:

= 4000 6000

vergleicht den Speicherinhalt ab $4000 mit dem ab $6000. Das erste nicht übereinstimmende Byte wird angezeigt, und der Vergleich wird abgebrochen.

Wenn Sie also ein Maschinenprogramm geschrieben und überarbeitet haben und Sie wissen nicht mehr genau, worin eigentlich der Unterschied zwischen der 76. und der 77. Version besteht, gehen Sie so vor: Laden Sie zuerst Version 76 und verschieben Sie diese mit dem »W«-Befehl in einen freien Speicherbereich. Laden Sie dann Version 77 und führen Sie den »=«-Befehl durch. Sofort finden Sie den Unterschied und können mit der Arbeit an Version 78 beginnen....

Wir wollen uns bei der Beschreibung der Trace-Befehle auf Anwendungsbeispiele konzentrieren. Zum Aufbau der Routine sei nur so viel gesagt: Gesteuert wird sie mit Hilfe des Prozessor-Interrupts, weil nur damit ein Eingriff ins laufende Maschinenprogramm möglich ist. Während des Trace-Ablaufs wird deswegen der Bildschirm kurzfristig aus- und eingeschaltet, weil alle anderen Interruptanforderungen wie zum Beispiel durch den Video-Chip, verhindert werden müssen. Da die Befehle eines Programms nicht nur angezeigt, sondern auch wirklich ausgeführt werden, ist der »SEI«-Befehl mit großer Vorsicht zu verwenden. Doch dazu später mehr. Wir wollen ein neues, besser geeignetes Beispiel verwenden als bisher. Tippen Sie also das folgende Miniprogramm mit dem Assembler ein (A 4000):

```
4000   LDA   #30    lade den Akku mit (ASCII-) 0
4002   JSR   FFD2   gib Akku auf dem Bildschirm aus
4005   CLC
4006   ADC   #01    erhöhe Akku um 1
4008   CMP   #39    vergleiche Akku mit (ASCII-) 9
400A   BCC   4002   springe, wenn Akku kleiner, zurück
400C   BRK          springe in SMON zurück
```

Starten Sie das Programm mit »G 4000«. Es muß die Zahlen von 0 bis 8 auf den Bildschirm schreiben.

◆ Trace-Stop

TS (Startadresse Stoppadresse)

Starten Sie nun unser Programm mit

TS 4000 4009

Die ersten Befehle werden ausgeführt (die Null ausgegeben, der Akku erhöht etc.), dann stoppt das Programm bei Adresse $4009 und springt in die Registeranzeige.

Genau genommen ist »TS« gar kein Trace-Befehl, das Programm läuft nämlich bis zur gewählten Stoppadresse in Echtzeit durch. Dort angekommen, können Sie die Register prüfen und gegebenfalls durch Überschreiben ändern. Mit »G«, »TW« oder »TB« (wird später erklärt) ohne weitere Adresseneingaben können Sie dann im Programmlauf fortfahren. SMON merkt sich nämlich, wo er stehen geblieben ist und arbeitet ab dieser Adresse weiter, wenn Sie nicht eine neue angeben.

Sinnvoll ist dieser Befehl immer dann, wenn in einem längeren Programm nur bestimmte Teile »getraced« werden sollen, der Anfang aber durchlaufen werden muß, um Variable zu setzen oder Benutzereingaben zu erfragen. Auch wenn man nicht ganz sicher ist, ob eine bestimmte Passage überhaupt jemals durchlaufen wird, kann man das mit »TS« überprüfen.

Zwei Einschränkungen gibt es allerdings wegen der Arbeitsweise dieses Befehls: SMON setzt im Programm an die Stoppadresse einen BRK-Befehl und merkt sich, welcher Befehl dort stand, um ihn wieder zurückzuschreiben. Deshalb funktioniert »TS« nur im RAM, nicht aber zum Beispiel im Basic oder im Betriebssystem. Auch darf die Speicherstelle, in der sich SMON den ausgetauschten Befehl merkt ($02BC) vom Programm nicht verändert werden, sonst ist eine korrekte Reparatur nicht mehr möglich.

Der wohl am häufigsten und vielseitigsten eingesetzte Trace-Befehl ist sicherlich »TW«.

◆ Trace Walk

TW (Startadresse)

Starten Sie unser Beispiel jetzt mit TW 4000

Der erste Befehl (LDA #30 in Adresse $4000) wird ausgeführt, SMON stoppt und zeigt dann die Inhalte aller Register in der gleichen Reihenfolge wie beim »R«-Kommando sowie den nächsten Befehl an. Im Akku steht jetzt 30, der Programmzähler zeigt auf $4002. Jetzt drücken Sie eine Taste. Der nächste Befehl (JSR FFD2) wird ausgeführt, der Programmzähler zeigt auf $FFD2. Achten Sie auf den Stackpointer: Sein Inhalt hat sich um 2 vermindert, weil der Prozessor auf dem Stack die Adresse abgelegt hat, an die er nach Beendigung der Subroutine zurückspringen soll. Der nächste angezeigte Befehl ist ein indirekter Sprung über $0326. Mit dem nächsten Tastendruck wird er durchgeführt.

Und so geht es munter weiter. Verzweifeln Sie nicht, wenn Sie auch nach den nächsten zehn Tastendrücken immer noch irgendwo im Betriebssystem »herumtracen« und von unserem Beispielprogramm weit und breit nichts mehr zu sehen ist. Ausnahmsweise ist unser Liebling einmal nicht im »Land der Träume« verschwunden, sondern tut, was er soll: Er arbeitet brav einen Befehl nach dem anderen alles ab, was zur Routine $FFD2 gehört, und das ist reichlich viel. Also bewegen Sie Ihre Finger, Sie haben's ja nicht anders gewollt. Irgendwann einmal, nach mehreren hundert gedrückten Tasten, befinden Sie sich plötzlich wieder in der Registeranzeige von SMON. Das Programm ist beendet. Nun werden Sie enttäuscht fragen, was man wohl mit einem Trace-Modus anfangen soll, der schon bei kleinsten Beispielprogrammen ein völlig undurchschaubares Chaos erzeugt? Nur Geduld, die Rettung naht in Gestalt der Taste »J«.

Falls ihre Hand noch nicht in Gips liegt, starten Sie das Ganze nochmal von vorn mit »TW 4000«. Diesmal drücken Sie aber jedesmal, wenn als nächster Befehl »JSR FFD2« angezeigt

wird, auf »J«. Der Effekt ist, daß die gesamte Subroutine auf einen Schlag abgearbeitet wird und Sie sofort wieder auf dem nächsten Befehl unseres Beispiels landen. Daß wir nicht gemogelt und die Befehle von »JSR FFD2« einfach unterschlagen haben, sehen Sie daran, daß der Akku tatsächlich auf dem Bildschirm ausgegeben worden ist (rechts neben FFD2). Jetzt können Sie unser Beispiel in aller Ruhe bis zu Ende durchgehen und verfolgen, wie der Akku erhöht wird, wie der Vergleich das Statusregister beeinflußt und wie dementsprechend der Rücksprung in die Schleife erfolgt.

Sie dürfen die »J«-Taste auch dann benutzen, wenn Sie schon mitten in der Subroutine sind. Aber hierbei ist äußerste Vorsicht geboten: Die Rücksprungadresse muß unbedingt oben auf dem Stack liegen, wenn Sie »J« drücken. Hat nämlich der Prozessor Werte auf dem Stack abgelegt (mit PHA oder PHP), dann erfolgt der Sprung irgendwo hin, nur nicht zurück ins Programm. Achten Sie deshalb genau auf die Anzeige des Stackpointers. Wenn dessen Wert genau so groß ist wie bei Beginn der Subroutine, kann nichts passieren. Sonst hilft nur noch der Reset-Taster, den Sie ja inzwischen hoffentlich eingebaut haben, oder eine ruhige Hand, die die Büroklammer an Pin 1 und 3 des User-Ports hält (Kostenpunkt der Reparatur bei Abrutschen zirka 100 Mark...)

»TW« bricht automatisch mit der Registeranzeige ab, wenn im Programm ein »BRK«-Befehl auftaucht. Wenn Ihnen das zu lange dauert oder Sie zwischendurch ein Register ändern möchten, können Sie den Trace-Modus jederzeit mit der Stopp-Taste verlassen. Anschließend können Sie wie bei »TS« beschrieben fortfahren.

Im Gegensatz zu »TS« können Sie mit »TW« auch im ROM herumstöbern; Sie haben es ja bei der Subroutine $FFD2 bereits getan. Einzige Einschränkung beim »TW«-Befehl: Ihr Programm darf keinen »SEI« enthalten, da dieser den Interrupt und damit auch den Trace-Modus lahmlegt. Verlassen Sie in diesem Falle »TW« mit STOP und starten erneut hinter dem »SEI«-Befehl. Allerdings müssen Sie in Kauf nehmen, daß das Programm normalerweise nicht mehr korrekt arbeitet.

Das nächste Programm soll als weiteres Beispiel für den TW-Modus dienen. Geben Sie es folgendermaßen ein:

5000	LDA	#00	lädt den Akku mit »0«
5002	TAX		überträgt den Akku ins X-Register
5003	.OC		ein mysteriöses Byte
5004	LDA	#04	lädt den Akku mit »4«
5006	TAY		überträgt den Akku ins Y-Register
5007	BRK		springt in SMON

Wenn wir dieses kleine Programm abarbeiten, müßte das X-Register auf »0« stehen, während Akku und Y-Register mit »4« geladen sind. Starten wir also das Programm mit »G 5000« und schauen uns die Register an.

Seltsamerweise enthalten alle Register eine »0«. Vorsichtig, wie wir sind, überschreiben wir die drei Register mit »FF«, um die Veränderung deutlich kontrollieren zu können.

Dann starten wir mit »G 5000« ein zweites Mal. Gegen alle Gesetze der Vernunft erscheint wieder das »falsche« Ergebnis — alle drei Register sind »0«. Hier soll uns jetzt der TW-Modus weiterhelfen, indem er uns zeigt, was in Wirklichkeit passiert.

Geben wir »TW 5000« ein. Der erste Befehl (LDA #00) ist durchgeführt, im Akku erscheint die Null. Jetzt steht der Programmzähler auf dem folgenden Befehl »5002 TAX«. Nach Drücken einer Taste wird dieser Befehl ausgeführt und es erscheint die Null im X-Register. Beim folgenden Befehl müssen wir feststellen, daß der Disassembler nicht in der Lage ist, ihn zu interpretieren — er gibt drei Sternchen aus. Hierbei handelt es sich um unser Byte »0C«.

Wieder ein Tastendruck; und dann erkennen wir, daß etwas Merkwürdiges passiert ist. Der Prozessor hat augenscheinlich den nächsten Befehl (LDA #04) übersprungen und steht schon auf dem folgenden »TAY«. So also wird unser Programm abgearbeitet. Damit ist auch das »falsche« Ergebnis erklärt. Bleibt nur noch die Frage nach dem Grund für dieses seltsame Verhalten. Und der ist sicherlich in dem mysteriösen Byte »OC« zu suchen. Hierbei handelt es sich um einen der »inoffiziellen« Opcodes (auch Pseudo-Opcodes genannt), die aufgrund der Prozessorarchitektur vorhanden sind und in manchen Programmen ihr Unwesen treiben — wie wir zu unserem Leidwesen erfahren mußten. Das Byte »OC« wirkt wie ein »NOP«, der eine Länge von 3 Byte hat. Deshalb wird der folgende 2-Byte-Befehl (LDA #04) verschluckt.

Es gibt noch einiges zu entdecken am 6502 und 6510 — TW macht's möglich.

Häufig ist es nicht sinnvoll, ein Programm von Anfang an im TW-Modus laufen zu lassen. Zum anderen sind gerade Schleifen, die per Hand mit »TW« durchlaufen werden müssen, eine ermüdende Angelegenheit. Hier bietet SMON neben dem bereits beschriebenen »TS« eine weitere Trace-Möglichkeit an:

◆ Trace Break

TB (Adresse Anzahl der Durchläufe)

◆ Trace Quick

TQ (Adresse)

Geben Sie als Beispiel folgendes Programm ein:

6000	LDY	#00	Y als Zähler auf »0«
6002	LDA	600E,Y	Werte von $600E ff. sollen geladen werden
6005	JSR	FFD2	Ausgabe der Zeichen auf dem Bildschirm
6008	INY		der Zähler wird erhöht
6009	CPY	#0E	Zähler schon »14«?
600B	BCC	6000	wenn nein, dann nächsten Wert holen
601D	BRK		

Bei $600E soll nun ein Text stehen, den das Programm ausgibt. Die einfachste Art, mit SMON Texte in den Speicher zu schreiben, besteht im »K«-Befehl. Geben Sie

K 600E

ein (danach natürlich Return) und drücken Sie die STOP-Taste. Fahren Sie mit dem Cursor an das erste ausgegebene Zeichen (vermutlich ein Punkt) und schreiben Sie — ohne Anführungszeichen:

»FEHLER BEHOBEN«

Drücken Sie dann Return, um die Zeile an den Rechner zu übergeben. Wenn Sie das Programm starten, werden Sie wieder einmal Gelegenheit haben, sich in Ruhe etwas zu trinken zu holen (Prost!), denn das Programm enthält einen dummen Fehler und beschäftigt den Computer für eine lange, lange Zeit. Genauer gesagt, bis Sie ihn mit Reset (zum Beispiel durch RUN/STOP-RESTORE) erlösen.

Nun soll SMON helfen, diesen Fehler zu lokalisieren. Setzen Sie zuerst einmal einen Breakpoint bei $6002 und begrenzen die Durchläufe auf die maximale Anzahl:

TB 6002 0E

und starten mit

TQ 6000

den Quicktrace bei $6000. Das Programm läuft so lange, bis zum 14. Mal die Adresse $6002 erreicht wird und springt dann in den TW-Modus. Wenn Sie sich jetzt die Registerinhalte genau anschauen, müßte ihnen der Fehler geradezu ins Auge springen. Wie groß sollte denn das Y-Register sein? Welchen Wert sollte der Akku haben? NA?! (Auflösung erfolgt mit der Bekanntgabe der Gewinner des großen Preisausschreibens aus dem letzten Heft...)

Wenn Sie eine Zeitlang mit SMON gearbeitet haben, werden Ihnen eventuell einige Voreinstellungen nicht gefallen. Besitzer einer Datasette zum Beispiel müssen jedesmal mit »I 01« auf ihre Geräteadresse einstellen. Wenn Sie sich an unserem

»Preisausschreiben« vom letzten Mal beteiligt haben, dürften Ihnen diese Speicherstellen bereits bekannt sein. Übrigens sind zirka 235.982 richtige Lösungen eingegangen; wir haben daraufhin ein Programm geschrieben, um den Gewinner zu ermitteln. Dieses läuft zur Zeit auf einem Großrechner in den USA, da der Speicher des C 64 nicht ausreichte. Wir werden Sie nach Abschluß des Programmlaufes, den wir für Mitte Juni dieses Jahres erwarten, informieren....

Sollten Sie nicht zu den glücklichen Gewinnern zählen, geben wir Ihnen im folgenden die Speicherstellen an, in denen die Voreinstellungen stehen. Diese können Sie dann mit dem »M«-Befehl Ihren Wünschen entsprechend abändern.

Vergessen Sie aber nicht, Ihre geänderte Version mit »S "@:SMON $C000" C000 CE09« abzuspeichern.

Hintergrund-Farbe	:	$C220
Schreibfarbe	:	$C228
Ger. Adr. Drucker	:	$C21B
Ger. Adr. F1/Cass	:	$C216

Wer steht wo ? — Wegweiser durch SMON

Wollen Sie Routinen in SMON analysieren oder verändern, müssen Sie die Einsprungadressen der einzelnen Befehle kennen. Diese lassen sich leicht finden:

Am Anfang des Programms (ab $C00B) befindet sich eine Liste aller Kommandos gefolgt von den Adressen (ab $C02B), bei denen diese Befehle beginnen. Lassen Sie sich mit »M C00B C02B« die Befehlsliste anzeigen (siehe Bild 1). Sie sehen, daß am Ende 5 Nullen stehen; hier können Sie eigene neue Befehle einbauen. Mit »M C02B C06B« erhalten Sie die Liste der Einsprungadressen, immer in der Reihenfolge Low-Byte — Hig-Byte, diesmal mit 10 Nullen am Ende, weil ja zu jedem Befehl 2 Byte für die Adresse gehören.

Wir wollen nun als Beispiel heraussuchen, wo die Routine zum Füllen eines Speicherbereichs mit einem vorgegebenen Byte steht. Der Befehl dazu ist »O«, also der 20. Befehl ($C01E). Die zugehörige Adresse ist ebenfalls die 20. in der Liste also $C9C0 (in Speicherstelle $C051/$C052). Die Routine beginnt allerdings immer erst ein Byte später, in unserem Fall also bei $C9C1.

Disassemblieren Sie jetzt diese Routine mit »C C9C1 C9D3«. Sie erhalten folgende Befehle (natürlich ohne Kommentare):

```
        JSR   C27A    holt zwei Adressen nach $FB/FC
                      und $FD/FE
        JSR   C28D    holt das gewünschte Byte in den
                      Akku
        LDX   #00     initialisiert X-Register als Zähler
LOOP    STA   (FB,X)  speichert den Akku-Inhalt in der er-
                      sten Adresse ab
        PHA           merkt sich den Akku-Inhalt
        JSR   C463    erhöht die Adresse und vergleicht
                      mit der Endadresse
        PLA           holt Akku-Inhalt zurück
        BCC   C9C9    wenn Ende nicht erreicht, dann
                      Sprung auf LOOP
        RTS
```

Wenn Sie nun zum Beispiel den »O«-Befehl nur zum Löschen verwenden möchten, könnten Sie die Routine so abändern, daß nur die beiden Adressen eingegeben werden müssen. Überschreiben Sie einfach den Befehl JSR C28D mit LDA #00 im Disassemblerlisting, disassemblieren Sie erneut mit »D C9C1 C9D3« und überschreiben Sie die drei Sternchen bei C9C6 mit einem »NOP«, um die entstandene Lücke (3-Byte-Befehl wurde durch 2-Byte-Befehl ersetzt) aufzufüllen. Nun können Sie Ihre veränderte Routine ausprobieren. Geben Sie zum Beispiel »O 5000 5200« ein und überzeugen Sie sich mit »M 5000 5200« von der korrekten Ausführung. Sie können auch die so veränderte Routine mit »W C9C1 C9D3 CE09« ans Programmende kopieren und sich einen neuen Befehl »E« (Erase) schaffen. Sie brauchen dann nur in die erste Null am Ende der Kommandotabelle »45« (für »E«) und in die ersten beiden Nullen der Adreßtabelle »08« und »CE« zu schreiben (für die Adresse $CE09).

Machen Sie sich ruhig einmal die Mühe, auch andere Routinen im SMON auf diese Art und Weise zu analysieren und zu ändern. Erstens macht es Spaß und zweitens können Sie SMON Ihren eigenen Wünschen anpassen. Vielleicht fallen Ihnen ja noch Routinen ein, die in SMON fehlen. Wir würden uns über Verbesserungsvorschläge freuen.

Das »Gedächtnis« von SMON

Wenn Sie Programme mit SMON untersuchen oder verändern wollen, müssen Sie noch wissen, welche Speicherstellen SMON verwendet. Es soll ja Monitorprogramme geben, die die Basic-Zeiger als Arbeitsspeicher benutzen, so daß ein Basic-Programm nach dem Rücksprung aus dem Monitor gelöscht ist. SMON tut so etwas nicht. Aber natürlich braucht er auch Speicherstellen, um sich Werte merken zu können. Damit Sie Konflikten von Anfang an aus dem Wege gehen können, sind die wichtigsten hier dargestellt.

In der Zeropage belegt SMON den Bereich von $00A4 bis $00B6. Dort stehen Systemvariable für die Kassettenspeicherung und die RS232-Schnittstelle. Diese werden nur während des Betriebs der Kassette oder von RS232 gebraucht, sind ansonsten aber frei. Außerdem werden die Speicherstellen $00FB bis $00FF benutzt, die sowieso zur freien Verfügung des Anwenders vorgesehen sind. Alle anderen Zeiger in der Zeropage, also insbesondere die Speicherverwaltung für Basic bleiben unbeeinflußt.

Als weiteren Arbeitsspeicher benutzt SMON den Bereich von $02A8 bis $02C0. Auch dieser Bereich wird vom Betriebssystem nicht benutzt, so daß keine Konflikte entstehen dürften. Beim Assemblieren wird zusätzlich noch der Kassettenpuffer als Speicher für die Labels benötigt. Dieser bleibt ansonsten aber auch unverändert; das ist wichtig, wenn Maschinenroutinen dort abgelegt werden sollen.

Alles in allem ist SMON also recht verträglich.

SMON verschieben? — Mit SMON!

Eine Reihe von Anfragen hat uns erreicht, ob man SMON nicht mit Hilfe des »W«-, »V«- oder »C«-Kommandos verschieben könne. Alle Versuche in dieser Richtung seien fehlgeschlagen. Einige Leser meinten auch, in der V-Routine müsse ein Fehler stecken. Diesmal sind wir jedoch völlig schuldlos; es gibt nämlich einige Befehle in SMON, die keine Sprungadressen sind und sich trotzdem auf den Bereich ($C000-) beziehen, in dem SMON steht.

Dazu gehören in erster Linie die oben erwähnten Einsprungadressen, deren High-Byte natürlich geändert werden muß, wenn SMON in einem anderen Speicherbereich laufen soll. Es gibt aber auch Befehle, die eine Adresse im Programm in einem Vektor ablegen müssen. Disassemblieren Sie einmal den Anfang von SMON mit »D C000 C00B«. Sie erhalten

```
LDA   #14    Low-Byte der BREAK-Routine von SMON
STA   0316   im Break-Vektor abspeichern
LDA   #C2    High-Byte (!) siehe oben
STA   0317   siehe oben
BRK
```

Damit wird der Break-Vektor des Betriebssystems auf den SMON gesetzt und mit dem anschließenden — und jedem weiteren BRK-Befehl — springt das Programm in SMONs BREAK-Routine. Wenn SMON in einem anderen Bereich als $C000

laufen soll, dann müssen diese Befehle geändert werden. Heraussuchen kann man sie mit »FIC*,C000 D000«. Sie wissen doch noch, was diese Anweisung bedeutet: Suche mir alle Befehle, die ein Register unmittelbar mit einem Wert laden, der mit $C beginnt. Aber Vorsicht! Nicht alles, was da angezeigt wird, muß auch geändert werden! Um Ihnen weitere Stunden sinnlosen Herumbrütens zu ersparen, wollen wir als Beispiel zeigen, wie man SMON in den Bereich $9000 bis $A000 verlegen kann. Natürlich geht das im Prinzip für jeden anderen Bereich genauso; wir selbst haben insgesamt fünf SMON-Versionen für fünf verschiedene Speicherbereiche, von denen eine immer paßt.

1. Wir verschieben zuerst das ganze Programm ohne Umrechnen in den neuen Bereich:
   W C000 CE09 9000
2. Nun lassen wir alle absoluten (3-Byte) Befehle umrechnen. Die Tabellen am Anfang von SMON bleiben verschont:
   V C000 CE09 9000 9214 9E09
3. Als nächstes ändern wir die High-Bytes der Befehlsadressen. Geben Sie
   M 902B 906B
   ein und ändern Sie in jedem zweiten Byte das »C« durch Überschreiben in »9«. Vergessen Sie nicht, am Ende jeder Zeile »Return« zu drücken, damit Ihre Änderung auch übernommen wird.
4. Als letztes bleiben noch die oben beschriebenen Direktlade-Befehle. Sie müssen so geändert werden, daß sie sich auf den neuen Bereich $9... beziehen. Suchen Sie sie mit
   FIC*,9000 9E09
   heraus. Sie erhalten

9005	LDA	#C2	ändern
9124	CPX	#C0	nicht ändern
9386	LDY	#C0	ändern
9441	CMP	#C0	nicht ändern
987F	LDX	#C3	nicht ändern
988D	LDX	#C1	nicht ändern
9992	LDA	#C1	nicht ändern
9C2C	LDA	#CC	ändern
9C5B	LDA	#C2	ändern
9CF4	LDA	#CC	ändern
9DA1	LDX	#CC	ändern
9E03	LDA	#CC	ändern

Ändern Sie nur in den gekennzeichneten Zeilen das »C« in »9« um. Sie sehen, es gibt keine Regel, welche Befehle zu ändern sind und welche nicht. Aus diesem Grunde müssen Sie diese Änderungen »von Hand« vornehmen.

Speichern Sie jetzt die fertige Version unbedingt mit »S "SMON $9000" 9000 9E09« ab. Nun starten Sie die 9000-Version mit »G 9000« oder von Basic aus mit SYS 36864 (SYS 9 x 4096). Löschen Sie dann den alten SMON mit »O C000 D000 00«. Nur so können Sie sicher sein, daß der verschobene SMON auch einwandfrei arbeitet und Sie nichts übersehen haben.

Probieren Sie nun alle Befehle durch. Sie müssen genauso arbeiten wie bisher. Vor allem können Sie jetzt auch Programme wie »DOS 5.1« oder »Turbo Tape« untersuchen, die im $C000-Bereich stehen. Achten Sie aber, wenn Sie »SMON $9000« von Basic aus benutzen, darauf, daß das Basic ihn nicht überschreibt. String-Variable werden nämlich von $A000 nach unten hin aufgebaut und bis $9E09 ist nicht viel Platz. Schützen Sie im Zweifelsfalle den Bereich, indem Sie nach dem Laden des SMON $9000 eingeben:

NEW : POKE 56,144 : POKE 55,0

Damit ist SMON vor Überschreiben geschützt. Das ist natürlich bei dem SMON $C000 nicht nötig, weil Basic in diesen Bereich nicht hineinkommt.

Zunächst wieder ein Geständnis: Leider hat uns der Druckfehlerteufel auch in der Dezember-Ausgabe wieder erwischt, aber der arme Kerl will ja schließlich auch seinen Spaß haben. Verzeihen Sie ihm und uns

— daß er im Listing in Zeile 150 ein REM eingeschummelt hat, das dort überhaupt nichts zu suchen hat. Es verhindert nämlich, daß die Checksumme geprüft wird und Eingabefehler erkannt werden.
— daß er auf Seite 62 fälschlich behauptet hat, die »LOAD«-Routine beginne bei $C84F. In Wirklichkeit beginnt Sie bei $C84E.
— daß er auf Seite 63 bei den Hinweisen in eigener Sache empfiehlt, man möge bei »... Error in 40« (oder in 70) die DATAs kontrollieren. Natürlich tritt der Fehler nicht in den REM-Zeilen, sondern in Zeile 140 oder 180 auf.
— daß er schließlich schon vorgearbeitet hat und in der Januar-Ausgabe sein schändliches Handwerk getrieben hat. Dort steht bei der Beschreibung des 16-Bit-Vergleichs, das Zero-Flag sei gesetzt, wenn beide Werte gleich seien. Das stimmt auch, aber Sie wissen ja selbst, wo der Teufel steckt, nämlich im Detail, und der Druckfehlerteufel macht da keine Ausnahme. Er verführt uns nämlich zu dem eigentlich logischen Schluß, man könne auf die beschriebene Art und Weise prüfen, ob zwei 16-Bit-Werte gleich sind. Und genau das geht nicht. Das Zero-Flag hat nämlich die unangenehme Eigenschaft, auch dann gesetzt zu sein, wenn der erste Wert bis zu 255 größer (!) ist als der zweite. Will man also zuverlässig auf Gleichheit testen, muß man die beiden Werte voneinander abziehen und nachsehen, ob das Ergebnis Null ist.

Hinweise zum Abtippen

Wir geloben Besserung und als Zeichen tätiger Reue haben wir Ihnen das Eintippen der DATAs leichter gemacht. An anderer Stelle in diesem Heft finden Sie das Programm »MSE«. Damit ist ein Vertippen nahezu ausgeschlossen, und schneller und bequemer geht's obendrein auch noch. Sie finden also auf der nächsten Seite nicht mehr den gewohnten Basic-Lader, sondern statt dessen eine Art Hexdump, das Sie mit »MSE« eintippen können. Wie das genau geht, erfahren Sie in dem zugehörigen Artikel. Nach Beendigung der Tipparbeit speichern Sie Ihr Werk ab. Laden Sie nun das SMON-Maschinenprogramm vom letzten Mal und starten Sie es mit SYS 49152. Hängen Sie den neuen Teil mit

»L "SMON TEIL 4"«

dahinter und speichern Sie das Ganze mit

S "@:SMON $C000" C000 CE09«

wieder ab. Sie haben dann das vollständige Programm (15 Blöcke auf der Diskette) zur Verfügung.

Wir hoffen, Ihnen mit SMON ein wirklich brauchbares Werkzeug an die Hand gegeben zu haben. Die vielen Zuschriften und Anrufe, die wir im Verlauf dieser Serie erhalten haben, bestätigen uns in dieser Auffassung. Viele Leser sind nach unserem Eindruck dabei, in die Maschinensprache einzusteigen und deren Möglichkeiten zu nutzen. Und genau das wollten wir erreichen.

Damit wären wir eigentlich am Ende dieser Serie angelangt und würden uns mit den üblichen guten Wünschen verabschieden. Aber die freien Bytes am Ende von SMON, immerhin von $CE09 bis $D000, also über 500 Byte, haben uns keine Ruhe gelassen. Als »Zugabe« werden wir Ihnen deshalb in der nächsten Ausgabe noch einen kleinen Diskettenmonitor vorstellen, der, in SMON eingebunden, dessen Möglichkeit nutzen kann.

(N. Mann/D. Weineck/gk)

Hinweise zu Bild 1

Alle Eingaben erfolgen in der hexadezimalen Schreibweise. In Klammern angegebene Adreßeingaben können entfallen. SMON benutzt dann sinnvolle vorgegebene Werte. Bei allen Ausgabe-Befehlen ist gleichzeitig die Ausgabe auf einen Drucker möglich. Dazu werden die Befehle geSHIFTet eingegeben.

Bild 1. SMON-Befehlsübersicht

— **A 4000 (Assembler)**
symbolischer Assembler (Verarbeitung von Labels möglich) Startadresse $4000

— **B 4000 4200 (Basic-DATA)**
erzeugt Basic-DATA-Zeilen aus Maschinenprogramm im Bereich von $4000 - $41FF

— **C 4010 4200 4013 4000 4200 (Convert)**
in ein Programm, das von $4000 - $4200 im Speicher steht, soll bei $4010 ein 3-Byte-Befehl eingefügt werden. Dazu wird das Programm ab $4010 - $4200 auf die neue Adresse $4013 verschoben. Alle absoluten Adressen, die innerhalb des Programmbereichs ($4000 - $4200) stehen, werden umgerechnet, so daß die Sprungziele stimmen.

— **D 4000 (4100) (Disassembler)**
disassembliert den Bereich von $4000 (- $4100) mit Ausgabe der Hex-Werte. Änderungen sind durch Überschreiben der Befehle möglich.

— **F (Find)**
findet Zeichenketten (F), absolute Adressen (FA), relative Sprünge (FR), Tabellen (FT), Zeropageadressen (FZ) und Immediate-Befehle (FI)

— **G (4000) (Go)**
startet ein Maschinenprogramm, das bei $4000 im Speicher beginnt.

— **I 01 (I/O-Gerät)**
stellt die Gerätenummer für Floppy (08 oder 09) oder Datasette (01) ein.

— **K A000 (A500) (Kontrolle)**
zum schnellen Durchsuchen des Bereichs von $A000 (-$A500) nach ASCII-Zeichen (32 Byte pro Zeile). Änderungen sind durch Überschreiben der ASCII-Zeichen möglich.

— **L (4000) (Load)**
lädt ein Maschinenprogramm an die richtige oder eine angegebene Adresse ($4000)

— **M 4000 (4400) (Memory-Dump)**
gibt den Inhalt des Speichers von $4000 (- $43FF) in Hex-Bytes und ASCII-Code aus. Änderungen sind durch Überschreiben der Hex-Zahlen möglich.

— **O 4000 4500 AA (Occupy)**
füllt den Speicherbereich von $4000 - $4500 mit vorgegebenem Byte ($AA) aus

— **P 02 (Printer)**
setzt Geräteadresse 2 für Drucker

— **R (Register)**
zeigt die Registerinhalte und Flags an. Änderungen sind durch Überschreiben möglich.

— **S "Test" 4000 5000 (Save)**
speichert ein Programm von $4000 - $4FFF unter dem Namen 'Test' ab

— **TW (4000) (Trace Walk)**
führt auf Tastendruck den jeweils nächsten Maschinenbefehl aus und zeigt die Registerinhalte an. Subroutinen können in Echtzeit durchlaufen werden (J'). Wird keine Startadresse eingegeben, beginnt 'TW' bei der letzten mit 'R' angezeigten Adresse.

— **TB 4010 05 (Trace Break)**
setzt einen Haltepunkt für den Schnellschrittmodus bei $4010. Der Schnellschrittmodus wird unterbrochen, nachdem $4010 zum 5. Mal erreicht worden ist.

— **TQ 4000 (Trace quick)**
Schnellschrittmodus, springt beim Erreichen eines Haltepunktes in den Einzelschrittmodus.

— **TS 4000 4020 (Trace stop)**
arbeitet ein Programm ab $4000 in Echtzeit ab und springt beim Erreichen von $4020 in die Registeranzeige. Von dort aus kann (nach evtl. Änderung der Register) mit »G« oder »TW« fortgefahren werden. »TS« arbeitet nur im RAM-Speicher.

— **V 6000 6200 4000 4100 4200 (Verschieben)**
ändert in einem Programm von $4100 - $41FF alle absoluten Adressen, die sich auf den Bereich von $6000 - $6200 beziehen, auf einen neuen Bereich, der bei $4000 beginnt.

— **W 4000 4300 5000 (Write)**
verschiebt den Speicherinhalt von $4000 -$42FF nach $5000 ohne Umrechnung der Adressen (zum Beispiel Tabellen)

— **X (Exit)**
springt aus dem Monitor-Programm ins Basic zurück

— **# 49152**
Dezimalzahl umrechnen

— **$ 002B**
4-stellige Hex-Zahl umrechnen

— **% 01101010**
8-stellige Binärzahl umrechnen

— **? 0344 + 5234**
Addition oder Subtraktion zweier 4-stelliger Hex-Zahlen

— **= 4000 5000 (Vergleich)**
vergleicht den Speicherinhalt ab $4000 mit dem ab $5000.

Listing 1. Der 4. Teil von SMON. Um dieses Programm einzutippen, verwenden Sie bitte den MSE, den Sie als Listing in diesem Heft finden.

```
PROGRAMM : SMON TEIL 4  CBF1 CE09
---------------------------------

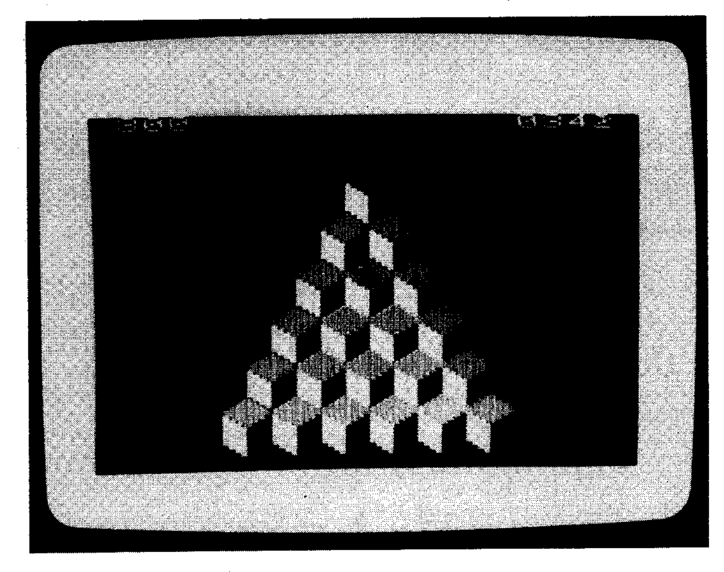
CBF1 : 68 68 20 CF FF C9 57 D0   DD
CBF9 : 03 4C 56 CD C9 42 D0 03   6A
CC01 : 4C D0 CD C9 51 D0 03 4C   A2
CC09 : 4F CD C9 53 F0 03 4C D1   18
CC11 : C2 20 8D C2 48 20 8D C2   E0
CC19 : 48 20 49 C2 A0 00 B1 FB   E5
CC21 : 8D BC 02 98 91 FB A9 36   AC
CC29 : 8D 16 03 A9 CC 8D 17 03   53
CC31 : A2 FC 4C EC C3 A2 03 68   30
CC39 : 9D 9A 02 CA 10 F9 68 68   48
CC41 : BA 8E AE 02 AD A8 02 85   61
CC49 : FC AD A9 02 85 FB AD BC   2F
CC51 : 02 A0 00 91 FB A9 14 8D   4E
CC59 : 16 03 A9 C2 8D 17 03 A9   A4
CC61 : 52 4C FF C2 20 51 C3 AD   28
CC69 : 11 D0 09 10 8D 11 D0 60   8C
CC71 : 8D AB 02 08 68 29 EF 8D   00
CC79 : AA 02 8E AC 02 8C AD 02   9D
CC81 : 68 18 69 01 8D A9 02 68   6F
CC89 : 69 00 8D A8 02 A9 80 8D   F5
CC91 : BC 02 D0 10 20 E5 CD 20   2D
CC99 : DD FD D8 A2 05 68 9D A8   5B
CCA1 : 02 CA 10 F9 AD 14 03 8D   EE
CCA9 : BB 02 AD 15 03 8D BA 02   FF
CCB1 : BA 8E AE 02 58 AD AA 02   40
CCB9 : 29 10 F0 08 20 65 CC A9   DB
CCC1 : 52 4C FF C2 2C BC 02 50   E3
CCC9 : 1F 38 AD A9 02 ED BD 02   2F
CCD1 : 8D B1 02 AD A8 02 ED BE   3D
CCD9 : 02 0D B1 02 D0 67 AD BF   8D
CCE1 : 02 D0 5F A9 80 8D BC 02   C4
CCE9 : 30 12 4E BC 02 90 CD AE   87
CCF1 : AE 02 9A A9 CC 48 A9 70   13
CCF9 : 48 4C BA CD 20 65 CC A9   83
CD01 : A8 85 FB A9 02 85 FC 20   20
CD09 : 4C C3 A0 00 B1 FB 20 2A   2F
CD11 : C3 C8 C0 07 F0 09 C0 01   A6
CD19 : F0 F2 20 4C C3 D0 ED AD   EA
CD21 : A9 02 AE A8 02 85 FB 86   D5
CD29 : FC 20 49 C3 20 CB C4 20   B4
CD31 : C7 C5 20 E4 FF F0 FB C9   8A
CD39 : 4A D0 0A A9 01 8D BC 02   17
CD41 : D0 2F CE BF 02 A5 91 C9   7B
CD49 : 7F D0 26 4C BD CC 20 F2   EC
CD51 : CD A9 40 D0 0A 20 F2 CD   26
CD59 : 08 68 8D AA 02 A9 80 8D   D9
CD61 : BC 02 BA 8E AE 02 20 49   AD
CD69 : C2 20 65 CC AD BC 02 F0   D9
CD71 : 37 A2 00 AD 11 D0 A8 29   3C
CD79 : 10 F0 10 98 29 EF 8D 11   83
CD81 : D0 EA EA A0 0C CA D0 FD   EB
CD89 : 88 D0 FA 78 A9 47 8D 04   5A
CD91 : DC 8E 05 DC AD 0E DC 29   A2
CD99 : 80 09 11 8D 0E DC A9 95   2D
CDA1 : A2 CC 8D BB 02 8E BA 02   08
CDA9 : AE AE 02 [illegible] 78 AD BB 02   6A
CDB1 : AE BA 02 8D 14 03 8E 15   AC
CDB9 : 03 AD A8 02 48 AD A9 02   9A
CDC1 : 48 AD AA 02 48 AD AB 02   6F
CDC9 : AE AC 02 AC AD 02 40 20   10
CDD1 : 8D C2 8D BE 02 20 8D C2   D7
CDD9 : 8D BD 02 20 8D C2 8D BF   6E
CDE1 : 02 4C D6 C2 AD B8 02 AE   1D
CDE9 : B9 02 8D 14 03 8E 15 03   88
CDF1 : 60 AD 14 03 AE 15 03 8D   48
CDF9 : B8 02 8E B9 02 A9 95 8D   6C
CE01 : 16 03 A9 CC 8D 17 03 60   FB
```

Q + Bert

Die VC 20-Version des bekannten Spielhallenhits. Nur besteht diese Variante aus drei verschiedenen Bildern.

Ziel des Spieles ist es, genau wie im bekannten Original, sämtliche Felder der Pyramide umzufärben. Das erreicht man, indem man einfach auf die noch unbeschränkten Felder springt. Damit es nicht zu einfach wird, ist noch eine Schlange mit im Spiel, mit der man nicht in Kontakt kommen darf. Wenn man es geschafft hat, alle Felder umzufärben, ohne die Schlange zu berühren, gelangt man in die nächste Pyramide. Nachdem auch die dritte Pyramide bewältigt ist, wechseln die Farben, und man fängt wieder mit der ersten Pyramide an, diesmal jedoch mit einem höheren Schwierigkeitsgrad.

Q + Bert wird mit folgenden Tasten gesteuert:
A — für einen Sprung nach links oben
F — für einen Sprung nach rechts oben
C — für einen Sprung nach links unten
Z — für einen Sprung nach rechts unten
Diese Steuerung ist etwas gewöhnungsbedürftig, eine Routine zur Abfrage des Joysticks hätte den Programmablauf jedoch stark verlangsamt. Außerdem können die Tasten in den Zeilen 320 bis 350 leicht geändert werden.



Das Spielfeld im ersten Bild

Bei den ersten beiden Pyramiden muß keine besondere Strategie angewendet werden. Zu beachten ist lediglich, daß man sich nur an der Schlange vorbeibewegen kann, nachdem sie aufgeblinkt hat. Weiter ist es hilfreich zu wissen, daß die Schlange sich nur dann bewegt, wenn sowohl deren X-Position als auch deren Y-Position nicht mit der eigenen X-Position beziehungsweise Y-Position übereinstimmt. In der dritten Pyramide kann man sich nicht an der Schlange vorbeimogeln. Aus diesem Grund muß man zuerst alle Felder umfärben, auf die die Schlange nicht gelangen kann und sie dann auf die umgefärbten Felder locken. Danach können die restlichen Felder umgefärbt werden.

Falls das Spiel so zu leicht erscheint, kann in Zeile 580 der Schwierigkeitsgrad erhöht werden, indem »S * 2« durch »S * 3« ersetzt wird.

(Oliver Pabst/ev)

Zeile	Funktion
10	Herabsetzen des für Basic verfügbaren Speicherplatzes
25-45	Erstellung des neuen Zeichensatzes
46-60	Festlegen der Variablen und Setzen der Farben
75	Aufruf zum Drucken der Pyramide
80-90	Initialisieren des Farb-RAMs (Modus für mehrfarbige Zeichen)
300-350	Tastaturabfrage
360-370	Tastaturpuffer entleeren / Schrittgeräusch
410-415	Fehlende Teile der Pyramide an die alte Position POKEn
420-421	Zeichen der Pyramide an der neuen Position ermitteln
424-430	Q+BERT am Rand der Pyramide?
440-445	Q+BERT POKEn
500-505	Feld schon umgefärbt?
520	Punktestand ausdrucken
530-560	Gesamte Pyramide umgefärbt?
580	Gegner bewegen? (RD kleiner 21-S*2)
600-630	Neue Position des Gegners errechnen
660-666	Fehlende Teile der Pyramide an die alte Position POKEn
670-675	Zeichen der Pyramide an der neuen Position ermitteln
676-677	Gegner am Rand der Pyramide?
678-680	Gegner POKEn
1000-1500	Erste Pyramide drucken
2000-2050	Feld umfärben / Punktestand erhöhen)
3000-3200	Zweite Pyramide drucken
4000-4200	Dritte Pyramide POKEn
5000-5310	Unterprogramme, die testen, ob die Bewegung des Gegners möglich ist, oder er von der Pyramide springt
8000-8050	Unterprogramm zum Abziehen der Punkte
10000-10200	Daten für die Umprogrammierung des Zeichensatzes
20000-20060	Fall von der Pyramide
20080-20100	Wackeln des Bildschirms
20200-20260	Abfrage, ob noch ein Spiel und Ausgabe der erreichten Punkte
30000-30020	Gegner fängt Q+BERT

Der Programmaufbau von »Q+Bert«

U	— Entscheidet, welche Pyramide gedruckt wird
RD	— Entscheidet, ob Gegner bewegt wird
S	— Farbe und Schwierigkeitsgrad
X	— Eigene X-Position
Y	— Eigene Y-Position
D	— X-Position des Gegners
DD	— Y-Position des Gegners
PU	— Anzahl der umgefärbten Felder
D1	— Zeichen der Pyramide unter dem Gegner
D2	— Zeichen der Pyramide unter dem Gegner
C1	— Zeichen der Pyramide unter Q+Bert
C2	— Zeichen der Pyramide unter Q+Bert
XX	— Eigene alte X-Position
YY	— Eigene alte Y-Position
S1	— Zeichen Q+BERT (links oben)
S2	— Zeichen Q+BERT (rechts oben)
AD	— Alte X-Position des Gegners
SD	— Alte Y-Position des Gegners
01	— Zeichen Gegner (links oben)
02	— Zeichen Gegner (rechts oben)
C3	— Zeichen der Pyramide unter Q+BERT (nur beim Fall)
C4	— Zeichen der Pyramide unter Q+BERT (nur beim Fall)
F1	— Hilfsfarbe
F2	— Farbe, in die umgefärbt wird

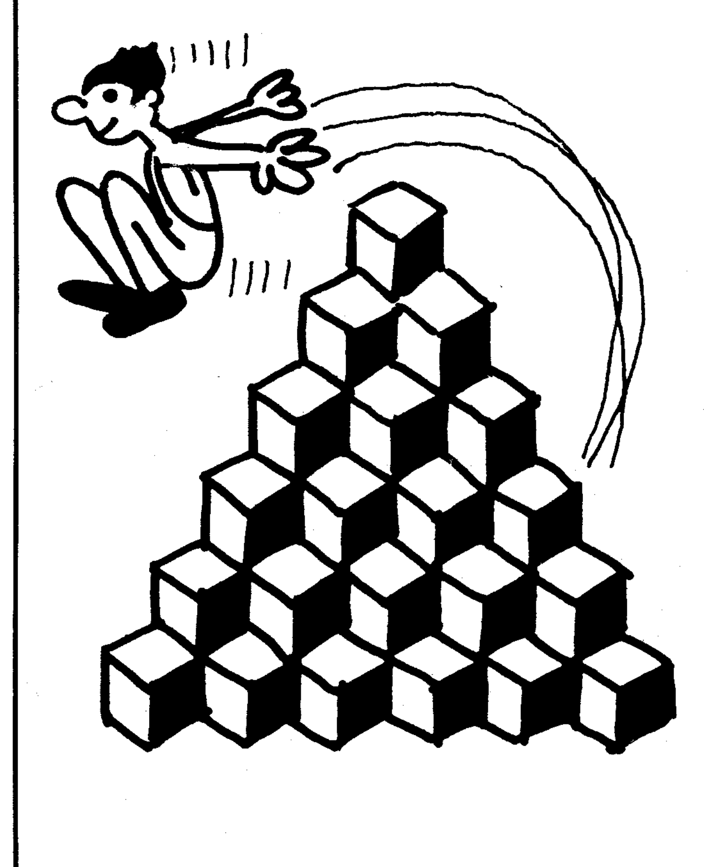
Variablentabelle

```
0 REM ======                                       <253>
1 REM Q+BERT                                       <057>
2 REM ======                                       <255>
3 REM OLIVER PABST                                 <221>
4 REM LUETTGENBERG 2                               <077>
5 REM 3320 SALZGITTER 1                            <150>
6 REM TEL. 05341/54845                             <222>
7 REM                                              <150>
10 POKE 52,24:POKE 56,24:CLR                       <062>
20 POKE 649,1                                      <171>
21 PRINT"{CLR}"                                    <133>
25 FOR I=6400 TO 6408:POKE I,0:NEXT I              <080>
30 FOR I=0 TO 175                                  <011>
40 READ A:POKE 6152+I,A:NEXT I                     <244>
45 FOR I=376 TO 456:POKE 6152+I,PEEK(32776+I)
   :NEXT:P=0                                       <129>
46 POKE 36869,254:U=1:TI$="000000":RD=15:S=3
                           <121>
50 POKE 36879,9                                    <063>
52 F1(3)=2:F2(3)=12:F1(4)=6:F2(4)=15:F1(5)=8
   :F2(5)=14                                       <059>
55 F1(6)=2:F2(6)=12                                <103>
60 POKE 36878,F1(S)*16 OR 15                       <083>
65 IF P=0 THEN 72                                  <014>
70 POKE 198,0:GOSUB 8000                           <106>
72 PRINT"{CLR}":RD=15                              <160>
75 ON U GOSUB 1000,3000,4000                       <094>
80 FOR I=38421 TO 38905:POKE I,11                  <035>
90 NEXT I                                          <037>
210 PU=0:D1=8:D2=7                                 <139>
220 TI$="000000":C1=8:C2=7                         <227>
230 POKE 198,0:IF U=3 THEN C1=1:C2=2               <011>
240 A$="":GOTO 310                                 <162>
300 GET A$:IF A$=""THEN 500                        <143>
310 XX=X:YY=Y                                      <232>
320 IF A$="Z"THEN X=X+1:Y=Y+66                     <025>
330 IF A$="A"THEN X=X-1:Y=Y-66                     <012>
340 IF A$="C"THEN X=X-1:Y=Y+66                     <023>
350 IF A$="F"THEN X=X+1:Y=Y-66                     <036>
360 POKE 198,0:POKE 36874,128                      <162>
370 POKE 36875,138:POKE 36874,0:POKE 36875,0
                           <082>
410 POKE 7680+XX+YY,C1:POKE 7680+XX+YY+1,C2
                           <053>
415 POKE 7680+XX+YY+22,3:POKE 7680+XX+YY+23,4
                           <250>
420 C1=PEEK(7680+X+Y)                              <184>
421 C2=PEEK(7680+X+Y+1)                            <149>
424 S1=15:S2=16                                    <029>
425 IF C1=1 THEN S1=11                             <203>
426 IF C2=2 THEN S2=12                             <208>
430 IF C1<>1 AND C1<>8 THEN 20000                  <155>
440 POKE 7680+X+Y,S1:POKE 7680+X+Y+1,S2            <017>
445 POKE 7680+X+Y+22,13:POKE 7680+X+Y+23,14
                           <024>
500 A=PEEK(38400+X+Y)AND 7                         <229>
505 IF A=3 THEN GOSUB 2000                         <161>
520 PRINT"{HOME,WHITE}";INT(P*25);"{HOME}
   "TAB(17);RIGHT$(TI$,4)                          <223>
530 IF PU=21 AND U=1 THEN U=2:GOTO 70              <074>
550 IF U=2 AND PU=37 THEN U=3:GOTO 70              <103>
560 IF U=3 AND PU=28 THEN U=1:S=S+1:GOTO 60
                           <220>
570 RD=RD+1                                        <245>
580 IF RD<21-S*2 THEN 800                          <152>
590 AD=D:SD=DD                                     <214>
600 IF X>D AND Y>DD THEN D=D+1:DD=DD+66
   :GOSUB 5000                                     <205>
610 IF X>D AND Y<DD THEN D=D+1:DD=DD-66
   :GOSUB 5100                                     <219>
620 IF X<D AND Y<DD THEN D=D-1:DD=DD-66
   :GOSUB 5200                                     <233>
630 IF X<D AND Y>DD THEN D=D-1:DD=DD+66
   :GOSUB 5300                                     <241>
660 POKE 7680+AD+SD,D1:POKE 7680+AD+SD+1,D2
                           <166>
666 POKE 7680+AD+SD+22,3:POKE 7680+AD+SD+23,4
                           <106>
667 IF Z=S1 THEN 30000                             <082>
670 D1=PEEK(7680+D+DD)                             <207>
675 D2=PEEK(7680+D+DD+1)                           <176>
676 O1=17:O2=18:IF D1=1 THEN O1=21                 <111>
677 IF D2=2 THEN O2=22                             <202>
678 POKE 7680+D+DD,O1:POKE 7680+D+DD+1,O2          <046>
680 POKE 7680+D+DD+22,19:POKE 7680+D+DD+23,20
                           <061>
690 RD=0                                           <044>
800 GOTO 300                                       <031>
1000 X=10:Y=9*22:D=10:DD=15*22                     <095>
1010 PRINT:PRINT                                   <097>
1020 PRINT"{10SPACE}AB"                            <095>
1030 PRINT"{10SPACE}CD"                            <110>
1040 PRINT"{10SPACE}EF"                            <124>
1050 PRINT"{9SPACE}AGHB"                           <013>
1060 PRINT"{9SPACE}CDCD"                           <019>
1070 PRINT"{9SPACE}EFEF"                           <037>
1080 PRINT"{8SPACE}AGHGHB"                         <186>
1090 PRINT"{8SPACE}CDCDCD"                         <184>
1100 PRINT"{8SPACE}EFEFEF"                         <206>
1110 PRINT"{7SPACE}AGHGHGHB"                       <103>
1120 PRINT"{7SPACE}CDCDCDCD"                       <093>
1130 PRINT"{7SPACE}EFEFEFEF"                       <119>
1140 PRINT"{6SPACE}AGHGHGHGHB"                     <020>
1150 PRINT"{6SPACE}CDCDCDCDCD"                     <002>
1160 PRINT"{6SPACE}EFEFEFEFEF"                     <032>
1170 PRINT"{5SPACE}AGHGHGHGHGHB"                   <193>
1180 PRINT"{5SPACE}CDCDCDCDCDCD"                   <167>
1190 PRINT"{5SPACE}EFEFEFEFEFEF"                   <201>
1200 PRINT"{5SPACE}IJIJIJIJIJIJ"                   <003>
1500 RETURN                                        <111>
2000 POKE 36875,180                                <064>
2005 PU=PU+1:P=P+1                                 <026>
2010 POKE 38400+X+Y,F2(S)                          <196>
2020 POKE 38400+X+1+Y,F2(S)                        <169>
2030 POKE 38400+X+Y+22,F2(S)                       <230>
2040 POKE 38400+X+23+Y,F2(S)                       <241>
2045 POKE 36875,0                                  <004>
2050 RETURN                                        <152>
3000 X=10:Y=6*22:D=10:DD=18*22                     <055>
3005 PRINT:PRINT                                   <052>
3010 PRINT"{9SPACE}ABAB"                           <142>
3015 PRINT"{9SPACE}CDCD"                           <155>
3020 PRINT"{9SPACE}EFEF"                           <168>
3030 PRINT"{2SPACE}ABAB{2SPACE}AGHGHB{2SPACE}
   ABAB                                            <073>
3040 PRINT"{2SPACE}CDCD{2SPACE}CDCDCD{2SPACE}
   CDCD                                            <087>
3050 PRINT"{2SPACE}EFEF{2SPACE}EFEFEF{2SPACE}
   EFEF                                            <125>
3060 PRINT" AGHGHBAGHGHGHBAGHGHB                   <044>
3070 PRINT" CDCDCDCDCDCDCDCDCDCD                   <010>
3080 PRINT" EFEFEFEFEFEFEFEFEFEF                   <061>
3090 PRINT" IJIJHGHGHGHGHGHGJIJ                    <123>
3100 PRINT"{4SPACE}CDCDCDCDCDCDCD                  <148>
3110 PRINT"{4SPACE}EFEFEFEFEFEFEF                  <186>
3120 PRINT"{4SPACE}IHGHGHGHGHGHGJ                  <228>
3130 PRINT"{5SPACE}CDCDCDCDCDCD                    <043>
3140 PRINT"{5SPACE}EFEFEFEFEFEF                    <077>
3150 PRINT"{5SPACE}IHGHGHGHGHGJ                    <115>
3160 PRINT"{6SPACE}CDCDCDCDCD                      <194>
3170 PRINT"{6SPACE}EFEFEFEFEF                      <224>
3180 PRINT"{6SPACE}IJIJIJIJIJ                      <018>
3200 RETURN                                        <026>
4000 PRINT:PRINT:X=10:Y=12*22:D=10:DD=6*22         <195>
4005 PRINT"{5SPACE}ABABABABABAB                    <129>
4010 PRINT"{5SPACE}CDCDCDCDCDCD                    <158>
```

Listing »Q+Bert«

```
4020 PRINT"{5SPACE}EFEFEFEFEFEF            <192>
4030 PRINT"{4SPACE}AGHGHGHGHGHGHB          <101>
4040 PRINT"{4SPACE}CDCDCDCDCDCDCD          <067>
4050 PRINT"{4SPACE}EFEFEFEFEFEFEF          <105>
4060 PRINT"{3SPACE}AGJIJIJIJIJIJIHB        <042>
4070 PRINT"{3SPACE}CD{12SPACE}CD           <190>
4080 PRINT"{3SPACE}EF{12SPACE}EF           <208>
4090 PRINT"{3SPACE}IHBABABABABABAGJ        <248>
4100 PRINT"{4SPACE}CDCDCDCDCDCDCD          <128>
4110 PRINT"{4SPACE}EFEFEFEFEFEFEF          <166>
4120 PRINT"{4SPACE}IHGHGHGHGHGHGJ          <208>
4130 PRINT"{5SPACE}CDCDCDCDCDCD            <023>
4140 PRINT"{5SPACE}EFEFEFEFEFEF            <057>
4150 PRINT"{5SPACE}IJIJIJIJIJIJ            <115>
4200 RETURN                                <006>
5000 Z=PEEK(7680+D+DD):IF Z>8 AND Z<>S1 THEN D=
   D-1:DD=DD-66                            <120>
5010 RETURN                                <051>
5100 Z=PEEK(7680+D+DD):IF Z>8 AND Z<>S1 THEN D=
   D-1:DD=DD+66                            <219>
5110 RETURN                                <151>
5200 Z=PEEK(7680+D+DD):IF Z>8 AND Z<>S1 THEN D=
   D+1:DD=DD+66                            <063>
5210 RETURN                                <252>
5300 Z=PEEK(7680+D+DD):IF Z>8 AND Z<>S1 THEN D=
   D+1:DD=DD-66                            <164>
5310 RETURN                                <096>
8000 PRINT"{WHITE,HOME,7SPACE}"            <084>
8005 FOR I=1 TO 2*VAL(TI$)                 <106>
8010 P=P-0.2                               <246>
8020 PRINT"{HOME}";INT(P*25)               <007>
8030 NEXT I                                <072>
8040 FOR I=1 TO 1800:NEXT                  <093>
8050 RETURN                                <031>
10000 DATA 2,2,10,10,42,42,170,170         <016>
10010 DATA 128,128,160,160,168,168,170,170 <202>
10020 DATA 170,170,106,106,90,90,86,86     <014>
10030 DATA 170,170,171,171,175,175,191,191 <222>
10040 DATA 85,85,85,85,85,85,85,85         <126>
10050 DATA 255,255,255,255,255,255,255,255 <000>
10060 DATA 149,149,165,165,169,169,170,170 <014>
10070 DATA 254,254,250,250,234,234,170,170 <250>
10080 DATA 85,85,21,21,5,5,1,1             <170>
10090 DATA 255,255,252,252,240,240,192,192 <022>
10100 DATA 2,2,15,15,47,44,188,191         <145>
10110 DATA 128,128,224,224,248,248,254,254 <052>
10120 DATA 175,175,111,111,90,90,86,86     <116>
10130 DATA 238,234,235,251,175,175,191,191 <072>
10135 DATA 254,254,251,251,239,236,188,191 <080>
10140 DATA 149,149,229,229,233,233,254,254 <086>
10150 DATA 254,254,240,240,196,192,160,168 <088>
10160 DATA 149,149,165,165,169,41,42,42    <211>
10170 DATA 168,170,106,96,82,80,86,86      <131>
10180 DATA 10,10,11,43,175,47,191,191      <107>
10190 DATA 2,2,0,0,4,0,160,168             <010>
10200 DATA 128,128,160,160,168,40,42,42    <233>
20000 FOR I=1 TO 21-Y/22                   <023>
20010 C1=PEEK(7680+X+Y+I*22)
   :C2=PEEK(7680+X+Y+1+I*22)               <186>
20015 C3=PEEK(7680+X+Y+22+I*22)
   :C4=PEEK(7680+X+Y+23+I*22)              <005>
20020 POKE 7680+X+Y+I*22,11                <132>
20021 POKE 7680+X+Y+1+I*22,12              <097>
20022 POKE 7680+X+Y+22+I*22,13             <150>
20023 POKE 7680+X+Y+23+I*22,14             <153>
20040 POKE 7680+X+Y+I*22,C1                <170>
20041 POKE 7680+X+Y+1+I*22,C2              <135>
20042 POKE 7680+X+Y+22+I*22,C3             <188>
20043 POKE 7680+X+Y+23+I*22,C4             <191>
20060 NEXT I                               <117>
20080 FOR I=1 TO 32                        <116>
20090 POKE 36865,38:POKE 36865,39:POKE 36865,40
   :POKE 36865,39                          <089>
20100 NEXT                                 <084>
20200 POKE 36879,27:PRINT"{CLR,BLUE}"      <060>
20210 POKE 36869,240                       <169>
20215 PRINT"{8DOWN,SPACE}IHRE PUNKTE:";INT(P*25)
                <135>
20220 PRINT"{2DOWN,SPACE}NOCH EIN SPIEL (J/N)"
                <226>
20230 GET A$:IF A$=""THEN 20230            <025>
20240 IF A$="N"THEN END                    <186>
20250 IF A$="J"THEN CLR:GOTO 46            <009>
20260 GOTO 20230                           <243>
30000 POKE 7680+D+DD,01:POKE 7680+D+DD+1,02
                <043>
30010 POKE 7680+D+DD+22,19:POKE 7680+D+DD+23,20
                <066>
30020 GOTO 20000                           <052>
```

Listing »Q+Bert« (Schluß)



Gehirntraining mit Supermemory

Den Schwierigkeitsgrad dieses kniffligen Denkspieles können Sie selbst bestimmmen. Doch Vorsicht! Es ist nicht so einfach wie es aussieht.

»Super Memory« ist ein Denkspiel für eine Person, die gegen sich selbst spielt. Aufgabe des Spielers ist es, Zahlen in einem 3 x 3-Kästchen großen Feld so zu ändern, daß sie zum Schluß alle den gleichen Wert haben. Die Schwierigkeit besteht nun darin, daß nur bestimmte Zahlengruppen verändert werden können. Diese insgesamt 9 Zahlengruppen können außerdem nur um einen Wert erhöht werden.

Wenn Sie das Programm starten, erscheint als erstes ein kleiner Vorspann mit der Anleitung zu dem Spiel. Nach dem Drücken einer Taste fragt Sie der Computer welche Zahl zum Schluß in den Kästchen stehen soll. Diese Zahl kann zwischen null und neun liegen. Angenommen Sie drücken nun die Zahl Acht (RETURN), fragt Sie der Computer als nächstes wieviele Schritte Sie minimal benötigen sollen, damit im Anzeigenfeld nur Achten stehen. Sie können nun einen bis unendlich viele Schritte angeben. Geben Sie nun als Beispiel eine Eins an (was die niedrigste Schwierigkeitsstufe bedeutet) erscheint nach einer ganz kurzen Denkpause des Computers das Spielfeld. Sie sehen nun in blauer Farbe die neun Zahlengruppen und die Buchstaben die ihnen zugeordnet sind. Der Buchstabe G bedeutet nun zum Beispiel, daß alle Zahlen in dem Feld, die dem Buchstaben G zugeordnet sind, um eins erhöht werden.

Da Sie bei dem Menüpunkt Schritte eine Eins angegeben haben, sind nun im Feld einige Siebener. Diese Siebener stehen in einem bestimmten Muster. Dieses Muster muß auch eines von den Blauen sein! Haben Sie dieses gefunden, drücken Sie die Taste die davor steht. Wenn die beiden Muster gleich waren, haben Sie gewonnen. Der Computer fragt Sie nun noch ob Sie noch ein Spiel spielen wollen. Spielen Sie noch eins! Tippen Sie nun bei »Anzahl der Schritte« eine Zwei ein, benötigen Sie schon zwei Schritte bis alle Zahlen im Feld gleich sind. Wie Sie sehen wird es nun bereits schwieriger. Durch Drücken der F1-Taste können Sie während des Spiels ein neues beginnen. Durch Drücken der F7-Taste können Sie bei einem laufenden Spiel alle Schritte zurücksetzen.

(Bernd Arnold/rg)

Listing »Super Memory«

```
0 GOTO 9                                        <194>
1 ************************                      <117>
2 *  SUPER MEM OR Y          *                  <209>
3 *  V ON BE RND ARNOLD      *                  <068>
4 *  SCHU LEN BURG STR.6     *                  <223>
5 *  8500 NUERNBERG 50    *                     <055>
6 *  TEL 804567           *                     <129>
7 ************************                      <123>
9 CLR:REM ALLE VARIABLEN LOESCHEN               <113>
10 POKE 53281,8:POKE 53280,7                    <062>
11 PRINT"[CLR,BROWN]"                           <016>
12 GOSUB 4500:REM SPRINGT ZUR EINLEITUNG        <071>
13 PRINT"[CLR,BROWN]":CLR                       <232>
14 INPUT"[DOWN]BITTE DIE ENDZAHL EINGEBEN ";A
   :AA=A:A=A+48                                 <036>
15 IF A<58 THEN GOTO 17:REM ZAHL AUF KORREKTHEI
   T UEBERPRUEFEN                               <059>
16 PRINT"[DOWN]DIE ZAHL DARF NICHT GROESSER
    ALS 9 SEIN":GOTO 10                         <104>
17 PRINT"[DOWN]BITTE DIE ANZAHL DER SCHRITTE
    EINGEBEN ":INPUT B                          <134>
20 A(1)=A:A(2)=A:A(3)=A:A(4)=A:A(5)=A:A(6)=A
   :A(7)=A:A(8)=A:A(9)=A:S=-1:B=B+1             <212>
40 GOTO 2000:REM ZUFALLSZAHLEN ERZEUGEN         <132>
90 REM BILDSCHIRMGRAFIK AUFBAUEN                <197>
100 PRINT"[CLR,2DOWN,6RIGHT,8DOWN,9RIGHT,WHITE]
   ";                                           <113>
110 PRINT CHR$(176)+CHR$(99)+CHR$(178)+CHR$(99)
   +CHR$(178)+CHR$(99)+CHR$(174);               <182>
120 PRINT"[DOWN,7LEFT]";:PRINT CHR$(98)+CHR$(A(
   1))+CHR$(98)+CHR$(A(2))+CHR$(98);            <244>
130 PRINT CHR$(A(3))+CHR$(125);
   :PRINT"[DOWN,7LEFT]";:PRINT CHR$(171)+CHR$(9
   9);                                          <046>
140 PRINT CHR$(123)+CHR$(99)+CHR$(123)+CHR$(99)
   +CHR$(179);:PRINT"[DOWN,7LEFT]";             <223>
150 PRINT CHR$(98)+CHR$(A(4))+CHR$(98)+CHR$(A(5
   ))+CHR$(98)+CHR$(A(6))+CHR$(98);             <039>
160 PRINT"[DOWN,7LEFT]";:PRINT CHR$(171)+CHR$(9
   9)+CHR$(123)+CHR$(99)+CHR$(123);             <235>
170 PRINT CHR$(99)+CHR$(179);
   :PRINT"[DOWN,7LEFT]";:PRINT CHR$(98)+CHR$(A(
   7));                                         <059>
180 PRINT CHR$(98)+CHR$(A(8))+CHR$(98)+CHR$(A(9
   ))+CHR$(98);:PRINT"[DOWN,7LEFT]";            <062>
190 PRINT CHR$(173)+CHR$(99)+CHR$(177)+CHR$(99)
   +CHR$(177)+CHR$(99)+CHR$(189)                <204>
200 PRINT"[BLUE,2RIGHT,4DOWN]F=[UP,RVSON,SPACE,
   DOWN,LEFT,SPACE,DOWN,LEFT,SPACE,UP,2SPACE,
   RVOFF,3RIGHT]H=[RVSON,2SPACE,UP,SPACE,DOWN,
   LEFT,SPACE,DOWN,LEFT,SPACE,RVOFF,3RIGHT,UP]
   G=[RVSON,3SPACE,UP,2LEFT,SPACE,2DOWN,LEFT,
   SPACE,RVOFF,UP,3RIGHT]";                     <017>
205 PRINT"T=[UP,RVSON,3SPACE,DOWN,2LEFT,SPACE,
   DOWN,LEFT,SPACE,RVOFF,UP,3RIGHT]";           <140>
210 PRINT"V=[DOWN,RVSON,3SPACE,UP,2LEFT,SPACE,
   UP,LEFT,SPACE]":PRINT"[10UP,2RIGHT]R=[UP,
   RVSON,3SPACE,DOWN,3LEFT,SPACE,DOWN,LEFT,
   SPACE,RVOFF,2DOWN]"                          <213>
220 PRINT"[2RIGHT]C=[DOWN,RVSON,3SPACE,3LEFT,UP,
   SPACE,UP,LEFT,SPACE,RVOFF,3UP,27RIGHT]Y=[UP,
   RVSON,3SPACE,DOWN,LEFT,SPACE,DOWN,LEFT,SPACE,
   RVOFF,3DOWN,2LEFT]";                         <079>
230 PRINT"[3LEFT]B=[UP,2RIGHT,RVSON,SPACE,DOWN,
   LEFT,SPACE,DOWN,3LEFT,3SPACE]"               <140>
240 RETURN                                      <126>
250 REM ********************************        <159>
1000 PRINT"[HOME]":S=S+1:REM S=SCHRITTE         <007>
1010 PRINT"[BROWN,SPACE]BITTE DAS FELD EINGEBEN,
    DAS SIE VER-[3SPACE]AENDERN WOLLEN"         <220>
1015 PRINT"[RIGHT,2DOWN]SCHRITTE:"S             <009>
1016 PRINT TAB(18)"[UP]ENDZAHL LAUTET "AA       <245>
1017 PRINT"[DOWN,RIGHT]F1=NEUES SPIEL"
   :PRINT" F7=ALLE SCHRITTE ZURUECK"            <054>
1020 GET A$:REM TASTATURABFRAGE                 <046>
```

Listing »Super Memory«

```
1025 IF A$="[F7]"THEN 4100:REM "[F7]" = F7 <046>
1030 IF A$="R"THEN 1140                    <175>
1050 IF A$="T"THEN 1150                    <198>
1060 IF A$="Y"THEN 1160                    <214>
1070 IF A$="F"THEN 1170                    <206>
1080 IF A$="G"THEN 1180                    <218>
1090 IF A$="H"THEN 1190                    <230>
1100 IF A$="C"THEN 1200                    <227>
1110 IF A$="V"THEN 1210                    <001>
1120 IF A$="B"THEN 1220                    <248>
1121 IF A$="[F1]"THEN 13:REM "[F1]" = F1   <033>
1122 FOR T=1 TO 9:IF A(T)=A THEN X=X+1
  :REM  UEBERPRUEFEN OB ALLE ZAHLEN GLEICH SIND
                        <025>
1123 NEXT T                                <061>
1124 IF X=9 THEN 3000:REM WENN JA DANN  GRADULA
  TION                                     <135>
1125 X=0                                   <163>
1130 GOTO 1020                             <186>
1135 REM  R-ZAHLEN UM 1 ERHOEHEN           <100>
1140 A(1)=A(1)+1:A(2)=A(2)+1:A(3)=A(3)+1
  :A(4)=A(4)+1:A(7)=A(7)+1                 <215>
1141 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48
  :REM WENN ZAHL GROESSER ALS 9 DANN AUF 0
                        <099>
1142 NEXT T:GOSUB 100:GOTO 1000            <044>
1143 REM  T-ZAHLEN UM 1 ERHOEHEN           <110>
1150 A(1)=A(1)+1:A(2)=A(2)+1:A(3)=A(3)+1
  :A(5)=A(5)+1:A(8)=A(8)+1                 <229>
1151 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <141>
1152 NEXT T:GOSUB 100:GOTO 1000            <054>
1153 REM  Y-ZAHLEN UM 1 ERHOEHEN           <125>
1160 A(1)=A(1)+1:A(2)=A(2)+1:A(3)=A(3)+1
  :A(6)=A(6)+1:A(9)=A(9)+1                 <243>
1161 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <151>
1162 NEXT T:GOSUB 100:GOTO 1000            <064>
1163 REM  F-ZAHLEN UM 1 ERHOEHEN           <116>
1170 A(1)=A(1)+1:A(4)=A(4)+1:A(5)=A(5)+1
  :A(6)=A(6)+1:A(7)=A(7)+1                 <001>
1171 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <161>
1172 NEXT T:GOSUB 100:GOTO 1000            <074>
1173 REM  G-ZAHLEN UM 1 ERHOEHEN           <127>
1180 A(2)=A(2)+1:A(4)=A(4)+1:A(5)=A(5)+1
  :A(6)=A(6)+1:A(8)=A(8)+1                 <015>
1181 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <171>
1182 NEXT T:GOSUB 100:GOTO 1000            <084>
1183 REM  H-ZAHLEN UM 1 ERHOEHEN           <138>
1190 A(3)=A(3)+1:A(4)=A(4)+1:A(5)=A(5)+1
  :A(6)=A(6)+1:A(9)=A(9)+1                 <029>
1191 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <181>
1192 NEXT T:GOSUB 100:GOTO 1000            <094>
1193 REM  C-ZAHLEN UM 1 ERHOEHEN           <143>
1200 A(1)=A(1)+1:A(4)=A(4)+1:A(7)=A(7)+1
  :A(8)=A(8)+1:A(9)=A(9)+1                 <043>
1201 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <191>
1202 NEXT T:GOSUB 100:GOTO 1000            <104>
1203 REM  V-ZAHLEN UM 1 ERHOEHEN           <172>
1210 A(2)=A(2)+1:A(5)=A(5)+1:A(7)=A(7)+1
  :A(8)=A(8)+1:A(9)=A(9)+1                 <057>
1211 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <201>
1212 NEXT T:GOSUB 100:GOTO 1000            <114>
1213 REM  B-ZAHLEN UM 1 ERHOEHEN           <162>
1220 A(3)=A(3)+1:A(6)=A(6)+1:A(7)=A(7)+1
  :A(8)=A(8)+1:A(9)=A(9)+1                 <071>
1221 FOR T=1 TO 9:IF A(T)=58 THEN A(T)=48  <211>
1222 NEXT T:GOSUB 100:GOTO 1000
  :REM             ZAHLEN IN GRAFIK ERNEUERN
                        <182>
1223 REM ******************************    <070>
1224 REM ZUFALLSZAHLEN ERZEUGEN            <163>
2000 PRINT"[CLR,HOME,DOWN,7RIGHT]EINEN MOMENT
  BITTE!"                                  <014>
2001 Z=INT(9*RND(1))+1:BB=BB+1             <182>
2002 IF BB=B THEN 4000:REM KONTROLLE OB GENUG Z
  UFALLSZAHLEN ERZEUGT SIND                <008>
2003 GOTO 2010                             <038>
2010 ON Z GOTO 2020,2030,2040,2050,2060,2070,
  2080,2090,2100                           <180>
2020 A(1)=A(1)-1:A(2)=A(2)-1:A(3)=A(3)-1
  :A(4)=A(4)-1:A(7)=A(7)-1                 <079>
2021 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <244>
2022 NEXT T:GOTO 2001                      <073>
2023 REM ******************************    <105>
2030 A(1)=A(1)-1:A(2)=A(2)-1:A(3)=A(3)-1
  :A(5)=A(5)-1:A(8)=A(8)-1                 <093>
2031 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <254>
2032 NEXT T:GOTO 2001                      <083>
2033 REM ******************************    <115>
2040 A(1)=A(1)-1:A(2)=A(2)-1:A(3)=A(3)-1
  :A(6)=A(6)-1:A(9)=A(9)-1                 <107>
2041 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <008>
2042 NEXT T:GOTO 2001                      <093>
2043 REM ******************************    <125>
2050 A(1)=A(1)-1:A(4)=A(4)-1:A(5)=A(5)-1
  :A(6)=A(6)-1:A(7)=A(7)-1                 <122>
2051 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <019>
2052 NEXT T:GOTO 2001                      <104>
2053 REM ******************************    <136>
2060 A(2)=A(2)-1:A(4)=A(4)-1:A(5)=A(5)-1
  :A(6)=A(6)-1:A(8)=A(8)-1                 <136>
2061 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <029>
2062 NEXT T:GOTO 2001                      <114>
2063 REM ******************************    <146>
2070 A(3)=A(3)-1:A(4)=A(4)-1:A(5)=A(5)-1
  :A(6)=A(6)-1:A(9)=A(9)-1                 <150>
2071 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <039>
2072 NEXT T:GOTO 2001                      <124>
2073 REM ******************************    <156>
2080 A(1)=A(1)-1:A(4)=A(4)-1:A(7)=A(7)-1
  :A(8)=A(8)-1:A(9)=A(9)-1                 <132>
2081 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <049>
2082 NEXT T:GOTO 2001                      <134>
2083 REM ******************************    <166>
2090 A(2)=A(2)-1:A(5)=A(5)-1:A(7)=A(7)-1
  :A(8)=A(8)-1:A(9)=A(9)-1                 <178>
2091 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <059>
2092 NEXT T:GOTO 2001                      <144>
2093 REM ******************************    <176>
2100 A(3)=A(3)-1:A(6)=A(6)-1:A(7)=A(7)-1
  :A(8)=A(8)-1:A(9)=A(9)-1                 <192>
2101 FOR T=1 TO 9:IF A(T)=47 THEN A(T)=57  <069>
2102 NEXT T:GOTO 2001                      <154>
2103 REM  SPIEL GEWONNEN : SCHLUSSGRAFIK   <191>
3000 PRINT"[CLR]"                          <051>
3010 PRINT" GRADULIERE SIE HABEN DAS PROBLEM
   IN"S:PRINT"[DOWN,SPACE]SCHRITTEN GELOEST"
                        <143>
3020 IF S<=BB-1 THEN 3040                  <232>
3030 GOTO 3060                             <051>
3040 PRINT"[DOWN,SPACE]DAS IST DIE GERINGSTE
   ANZAHL VON"                             <204>
3050 PRINT"[DOWN,SPACE]SCHRITTEN DIE MOEGLICH
   WAR!"                                   <188>
3060 PRINT"[3DOWN,SPACE]WOLLEN SIE NOCH EIN
   SPIEL";                                 <125>
3070 INPUT A$:IF A$="J"THEN GOTO 13        <241>
3080 PRINT"[CLR,DOWN,RIGHT]SCHADE":END     <020>
4000 FOR T=1 TO 9:B(T)=A(T):NEXT T         <013>
4010 GOSUB 100:GOTO 1000                   <091>
4100 FOR T=1 TO 9:A(T)=B(T):NEXT T         <114>
4110 S=-1:GOSUB 100:GOTO 1000              <219>
4500 REM TITELBILD ERZEUGEN                <054>
4510 PRINT"[CLR,11DOWN,12RIGHT]";          <113>
4520 PRINT"BERND[2SPACE]ARNOLD[2DOWN,12LEFT]
  PRESENTIERT"                             <148>
4530 FOR T=0 TO 2400:NEXT T                <244>
4540 PRINT"[CLR,3DOWN,RIGHT,9DOWN,12RIGHT]";
                        <189>
4550 PRINT"SUPER MEMORY"                   <028>
4560 FOR T=0 TO 2400:NEXT T                <018>
4570 PRINT"[CLR,DOWN,RIGHT]64'ER[28SPACE]64'ER"
                        <217>
4580 PRINT"[10DOWN,14RIGHT]";              <077>
4590 PRINT"PRODUZIERT[DOWN,10LEFT]20.10.1984"
                        <028>
4600 PRINT"[9DOWN,RIGHT]64'ER[28SPACE]64'ER"
                        <236>
4610 FOR T=0 TO 2400:NEXT T                <069>
4620 PRINT"[CLR]"                          <142>
4630 PRINT"DIES IST EIN DENKSPIEL FUER JEDES
   ALTER."                                 <216>
4640 PRINT"[DOWN,SPACE]SINN DES SPIELES IST
   ES, ZAHLEN IN[DOWN,6SPACE]
  EINEM 3*3 KAESTCHEN ";                   <022>
4650 PRINT"GROSSEN FELD SO ZU[DOWN,2SPACE]
   MANIPULIEREN DASS ZUM SCHLUSS ALLE";    <229>
```

```
4660 PRINT" DEN[SPACE,DOWN,SPACE]GLEICHEN WERT
     HABEN.[DOWN]"                           <041>
4670 PRINT" DURCH DRUECKEN SPEZIELLER TASTEN
     DIE[DOWN,4SPACE]SPAETER ANGEZEIGT WERDEN,";
                              <131>
4680 PRINT" KOENNEN BE-[DOWN,3SPACE]STIMMTE
     ZAHLEN UM EINS ERHOEHT WERDEN.[DOWN]"  <211>
4690 PRINT" WELCHE ZAHLEN DAS SIND KOENNEN SIE
     AN[2SPACE,DOWN,SPACE]DEN MUSTERN ERKENNEN."
                              <217>
4700 PRINT"[2DOWN,4RIGHT]WEITER MIT RETURN"
                              <123>
4710 GET A$:IF A$<>CHR$(13)THEN 4710        <251>
4720 PRINT"[CLR,DOWN]"                       <003>
4730 PRINT" SIE KOENNEN WAEHLEN WELCHE ZAHL
     ZUM[SPACE,DOWN,4SPACE]SCHLUSS EINMAL IM
     ";                                      <252>
4740 PRINT"KAESTCHEN STEHEN[DOWN,6SPACE]SOLL.
    [DOWN]"                                  <090>
4750 PRINT" AUSERDEM KOENNEN SIE WAEHLEN WIEVIE
    LE[SPACE,DOWN,2SPACE]SCHRITTE SIE MINDESTENS
     ";                                      <173>
4760 PRINT"DAZU BENOETI-[DOWN,3SPACE]GEN." <007>
4770 PRINT"[3DOWN,SPACE]PS: WENN IHNEN DAS SPIE
    L SCHWER ER-[5SPACE]SCHEINEN SOLLTE, "; <183>
4780 PRINT"MACHT NICHTS ES IST[4SPACE]NOCH KEIN
     MEISTER VOM HIMMEL GEFALLEN!"           <206>
4790 PRINT"[2DOWN,SPACE]ZUM SPIELEN BITTE TASTE
     DRUECKEN"                               <029>
4800 GET A$:IF A$=""THEN 4800               <107>
4810 RETURN                                  <106>
```

Listing »Super Memory« (Schluß)

Programmbeschreibung

0—11	Variablen — Bildschirm löschen, Cursorfarbe auf Braun, Bildschirmrahmen auf Gelb, Hintergrundfarbe auf Orange
13—40	Nach Endzahl und Schrittzahl fragen, Variablen für Anzeigefeld definieren, Sprung zu Zufallszahlen erzeugen
100—240	Bildschirm löschen, Anzeigefeld aufbauen (dies wird auch als Unterprogramm benützt)
1000—1017	Schritte — Endzahl anzeigen, F-Tasten erklären
1020—1121	Tastaturabfrage
1122—1130	Prüfen ob alle Zahlen im Feld gleich sind, wenn »ja« dann Spiel zu Ende, wenn »nein« erneut Tastaturabfrage
1135—1222	Unterprogramme: Die verschiedenen Zahlen werden nach Tastendruck um eins erhöht, Sprung danach zu »Anzeigefeld aufbauen«, damit dort die Zahlen erneuert werden, Sprung danach zur Tastaturabfrage
1224—2010	Zufallszahlen erzeugen, bestimmen welche Zufallszahlen erzeugt werden, wieviele; sind genug erzeugt, dann Sprung zum Unterprogramm, das sich die Zahlen merkt, damit sie nach Drücken der F7-Taste wieder erscheinen
2020—2102	Unterprogramme für die einzelnen Zufallszahlen, überprüfen ob einzelne Zahlen größer als neun sind, wenn ja, dann werden daraus Nuller
2103—3080	Spiel gewonnen, Schlußgrafik, Anzahl der Schritte ausgeben, fragen ob noch ein Spiel, wenn nein, Ende
4000—4010	Unterprogramm Zufallszahlen merken, Sprung zu »Anzeigefeld aufbauen, Sprung zu Tastaturabfrage«
4100—4110	Unterprogramm »Schritte« zurücksetzen, Sprung zu »Anzeigefeld aufbauen«, Sprung zu »Tastaturabfrage«
4500—4610	Titelbild erzeugen, auf Copyright hinweisen
4620—4710	Erste Anleitungsseite erzeugen, nach Drücken von RETURN fortfahren
4720—4810	Zweite Anleitungsseite erzeugen, nach Tastendruck Spiel beginnen

Tips & Tricks gesucht

Jeder Computer und jedes Programm hat seine speziellen Schwachstellen und Unzulänglichkeiten. Allerdings ist kaum ein Programmierer oder Anwender auf Dauer bereit, sich damit abzufinden. Wo auch sorgfältigste Lektüre von Handbüchern nicht weiterhilft, da wird so manche Stunde experimentiert, um eine Lösung zu finden (die oft in einer Basic-Zeile Platz hat).

Wir suchen solche Tips und Tricks, um sie unseren Lesern zugänglich zu machen. Schließlich ist es wenig sinnvoll, sich wochenlang mit Problemen herumzuschlagen, die andere bereits gelöst haben.

Wenn Sie also interessante Tips für den Umgang mit Computer, Floppy, Drucker oder sonstiger Hardware haben, wenn Sie bei kommerzieller Software einige Kniffe kennen, die nicht in der Anleitung stehen, oder wenn Sie interessante Problemlösungen statt in vier Seiten Listing in ein oder zwei Basic-Zeilen untergebracht haben, dann sollten Sie uns auf jeden Fall einmal schreiben.

Bitte geben Sie genau den Computertyp und die Gerätekonfiguration oder die Software an, und senden Sie Ihren Tip oder Trick an die
Redaktion 64'er
Markt & Technik Verlag
Aktiengesellschaft
Hans-Pinsel-Str. 2
8013 Haar bei München

Wir suchen

Anwendung des Monats, was ist das? Nun, Sie haben einen Commodore 64 oder einen VC 20 und versuchen diesen irgendwie sinnvoll einzusetzen. Unter einer sinnvollen Anwendung versteht die 64'er Redaktion alles, was beispielsweise Programme im häuslichen Bereich bewirken. Es kann sich dabei um die Berechnung der Benzinkosten für Ihren Wagen handeln, um ein eigenes Textverarbeitungsprogramm gehen, sich um die Verwaltung Ihrer Tiefkühltruhe drehen oder ein ausgeklügeltes Telefon- und Adreßregister sein.

Setzen Sie Ihren VC 20/C 64 mehr oder weniger beruflich ein? Auch, oder vor allem, das ist eine sinnvolle Anwendung. Sie führen die Lohn- und Gehaltsabrechnung, Ihre Lagerverwaltung, die Bestellungen auf einem Commodore-Heimcomputer durch? So spezielle Anwendungen wie die Berechnung der Statik von selbstgezimmerten Regalen, von Klimadiagrammen oder Vokabellernprogrammen für den Schulunterricht oder die Zinsberechnung bei Krediten sind ebenfalls Themen, die mehr als konkurrenzfähig sind.

Uns ist die Anwendung des Monats

500

wert.

Schreiben Sie uns, was Sie mit Ihrem Computer machen:

Redaktion 64'er, Aktion: Anwendung des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Cursorsteuerung leicht gemacht

Bei professionellen Programmen der PC-Klasse kann der Cursor meist über Eingabegeräte wie die Maus positioniert werden. Daß es auch recht gut mit dem Joystick und dem C 64 funktioniert, beweist dieses Programm.

Haben Sie den kurzen MSE-Lader eingetippt und gestartet, können Sie den Cursor mit einem Joystick in Port 2 steuern. Das Steuerprogramm befindet sich ab Adresse $C000 bis $C066 im Speicher. Die ersten 15 Bytes nimmt eine Initialisierungsroutine in Anspruch, die den Interruptvektor auf $C00F legt und das Steuerprogramm in den Kernalinterrupt einbindet. Die Routine wird mit SYS 49152 aufgerufen. Das Programm benutzt Speicherzelle $02 als Zählregister, da der Joystick nur bei jedem sechsten Interrupt abgefragt wird. In Zeile 300 des Assemblerlistings wird die Zählvariable um 1 erniedrigt. Ist die Variable 0, wird sie auf 6 gesetzt und in die Steuerungsroutine verzweigt. Ab Zeile 360 wird der Tastaturpuffer auf freien Platz überprüft. Sollte der Puffer voll sein, wird sofort in die Interruptroutine des Betriebssystems ($EA31) gesprungen. In Zeile 400 wird der Joystick abgefragt und das entsprechende Cursorsteuerzeichen in den Akku geladen. Ab 800 wird das Zeichen in den Tastaturpuffer geschrieben und der Pufferzeiger erhöht.

(P.Siepen/hm)

Listing zu »Cursorsteuerung leicht gemacht«. Das Programm muß mit dem MSE auf Seite 68 eingegeben werden.

```
 PROGRAMM : CURSOR                     C000 C066
------------------------------------------------
C000 : A9 0F 8D 14 03 A9 C0 8D   B2
C008 : 15 03 A9 06 85 02 60 C6   41
C010 : 02 F0 03 4C 31 EA A9 06   F2
C018 : 85 02 A6 C6 E0 08 90 03   B7
C020 : 4C 31 EA AD 00 DC 29 01   03
C028 : C9 00 D0 05 A9 91 20 5E   2A
C030 : C0 AD 00 DC 29 02 C9 00   2C
C038 : D0 05 A9 11 20 5E C0 AD   6B
C040 : 00 DC 29 04 C9 00 D0 05   63
C048 : A9 9D 20 5E C0 AD 00 DC   C7
C050 : 29 08 C9 00 D0 05 A9 1D   06
C058 : 20 5E C0 4C 31 EA A6 C6   F4
C060 : 9D 77 02 E6 C6 60 00 A0   C7
```

Assemblerprogramm der Cursorsteuerung

```
10 REM************************************
20 REM*    CURSOR STEUEREUNG               *
30 REM*                                    *
65 REM*       PETER SIEPEN                 *
70 REM*                                    *
75 REM*     VON-STEPHAN-STR.6              *
80 REM*                                    *
82 REM*     4200 OBERHAUSEN 1              *
84 REM*                                    *
85 REM*  TELEFON :(0208)/26555             *
90 REM************************************
95 :
100 SYS9*4096
110 .OPT P,OO
111 :
112 :
113 :
114 :
120 *= $C000
125 :
130 :
140 :
150 TEST   = $02          ;ZAEHLVARIABLE
160 JOY    = 56320        ;PORT #2
170 AZITP  = $C6          ;ANZAHL ZEICHEN IM PUFFER
174 :
175 :
180 LDA #<BEGINN;INTERUPTVEKTOR
185 STA $314
190 LDA #>BEGINN;AUF NEUE
195 STA $315     ;ADDRESSE SETZEN
200 LDA #$06     ;ZAEHLVARIABLE
205 STA TEST     ;HOCHSETZEN
210 RTS          ;ZURUECK ZU BASIC
215 :
220 :
225 :
300 BEGINN     DEC TEST
320 :          BEQ START     ;GENUG LEER IRR
330 :          JMP $EA31     ;NEIN WEITER MIT IRR
340 START      LDA #$06      ;ZAEHLVARIABLE HOCHSETZEN
350 :          STA TEST
360 :          LDX AZITP     ;TASTATURPUFFER
370 :          CPX #$08      ;VOLL
380 :          BCC WEITER
390 :          JMP $EA31     ;JA WEITER MIT INTERUPT
400 WEITER     LDA JOY       ;WENN JOY NICHT
410 :          AND #1        ;NACH OBEN
420 :          CMP #0        ;WEITER
430 :          BNE NOBEN
440 :          LDA #"□"      ;WENN JA STEUERZEICHEN
450 :          JSR AUSG      ;AUSGEBEN
460 NOBEN      LDA JOY
470 :          AND #2
480 :          CMP #0
490 :          BNE NUNTEN
500 :          LDA #"▧"
510 :          JSR AUSG
520 NUNTEN     LDA JOY
530 :          AND #4
540 :          CMP #0
550 :          BNE NLINKS
560 :          LDA #"▌"
570 :          JSR AUSG
580 NLINKS     LDA JOY
590 :          AND #8
600 :          CMP #0
700 :          BNE NRECHTS
710 :          LDA #"▐"
720 :          JSR AUSG
730 NRECHTS    JMP $EA31
740 :
750 :
760 :
770 :
780 :
790 :
800 AUSG       LDX $C6       ;X-REG FUER
810 :          STA $277,X    ;INDIZIERTE
820 :          INC AZITP     ;ADDRESSIERUNG
830 :          RTS           ;LADEN
840 :                        ;IN TASTPUFFER
850 :                        ;SCHREIBEN
860 :                        ;ANZAHL ZEICHEN
870 :                        ;IM TASTPUFFER
880 :                        ;ERHOEHEN
890 :
900 :
READY.
```

Basic-Zeilen genau betrachtet

Wenn Sie eine Basic-Zeile eingeben und anschließend RETURN drücken, legt der Interpreter die Zeile im Speicher ab. Will man Basic-Zeilen durch ein Maschinenprogramm erzeugen oder in einem Basic-Programm nach bestimmten Befehlen suchen, muß man wissen, wie und wo Basic-Zeilen gespeichert sind.

In den Speicherzellen 43/44 steht die Anfangsadresse des Basic-Speichers. Die Adresse, bei der Basic beginnt, erfahren Sie durch die Abfrage PRINT PEEK(43)+256*PEEK(44). Sie erhalten den Wert 2049. Das ist die Adresse, ab der die erste Basic-Zeile vom Interpreter gespeichert wird. Wie Basic-Zeilen im Speicher vorliegen, soll am folgenden Beispielprogramm erläutert werden.

```
10 Print "PROBE"
300 REM C - 64
1200 A$ = "A = A"
50000 END
```

Bild 1 und 2 zeigen, wie der Interpreter diese vier Basic-Zeilen ab der Adresse 2049 speichert. Die Werte in Klammern sind die Startadressen der einzelnen Basic-Zeilen und dienen nur dem besseren Überblick. Alle Werte sind dezimal und hexadezimal angegeben.

Die ersten beiden Bytes jeder Zeile heißen Linkpointer oder einfach Verbindungszeiger, manche sprechen auch von Koppeladressen. Der Wert dieser beiden Bytes entspricht immer der Startadresse der nächsten Zeile. Das erste Byte ist das Low- und das zweite das High-Byte der Adresse. In Bild 1 ergeben die ersten 2 Byte der ersten Zeile den Wert 15+8x256=2063, also die Adresse, bei der die zweite Zeile beginnt. Der Linkpointer der vierten Zeile (47+8x256) zeigt auf die Speicherstelle 2095. Die beiden Null-Codes in 2095 und 2096 (der Linkpointer der fünften Zeile) signalisieren dem Interpreter das Programmende.

Byte 3 und 4 jeder Zeile ergeben die Zeilennummer der jeweiligen Basic-Zeile. In der ersten Zeile ergeben diese 2 Byte 10 + 0 x 256 = 10.

Im Beispielprogramm steht, nach dem Linkpointer und der Zeilennummer, als fünftes Byte, das Token für PRINT (99). Um Speicherplatz zu sparen und eine schnelle Programmabarbeitung zu erzielen, werden alle Basic-Befehle (PRINT, REM, END,...) in nur 1 Byte übersetzt. Ein 1-Byte-Befehlswort heißt Token. Der Interpreter holt sich die Information, welches Token einem bestimmten Befehlswort entspricht, aus einer Zuordnungstabelle im ROM. Die Tabelle beginnt ab Adresse 41118 ($A090). Bild 3 zeigt einen Überblick über alle Basic-Befehle und deren Token. Die Leerstelle nach der Zeilennummer, die nach dem LIST-Befehl am Bildschirm zu sehen ist, wird nicht berücksichtigt. Die Leerstelle zwischen PRINT und dem Anführungszeichen steht als ASCII-Code 32 im sechsten Byte. Auf die gleiche Weise werden das Anführungszeichen, das Wort PROBE und das Schlußzeichen im ASCII-Code gespeichert. Das Beispiel zeigt, daß der Interpreter alle Zeichen, außer Befehlswörtern, im ASCII-Code speichert. Das gilt auch für Befehlswörter in Anführungszeichen ("PRINT"). Diese Unterscheidung können Sie in der dritten Zeile deutlich erkennen. Das erste Gleichheitszeichen stuft der Interpreter als Operator ein (Token 178, $B2), das Gleichheitszeichen in Anführungsstrichen als ASCII-Code 61 ($3D). Eine Aufstellung der ASCII-Codes finden Sie Ihrem Handbuch zum C 64 auf Seite 135.

Das Ende jeder Programmzeile ist durch eine Null markiert. Der Interpreter erkennt daran das Zeilenende und nimmt sich die nächste Zeile vor. Enthalten die beiden Bytes für den Linkpointer Null-Codes, ist für den Interpreter das Programm zu Ende. Im Beispiel sind es die Adressen 2095 und 2096 ($082F, $0830).

	Link-pointer		Zeilen-nummer		Basic-Text
(0801)	0F	08	0A	00	99 20 22 50 52 4F 42 45 22 00
(2049)	15	8	10	0	153 32 34 80 82 79 66 69 34 0
					Print " P R O B E "
(080F)	1C	08	2C	01	8F 20 43 20 2D 20 36 34 00
(2063)	28	8	44	1	143 32 67 32 45 32 54 52 0
					REM C — 6 4
(081C)	29	08	B0	04	41 24 B2 22 41 3D 41 22 00
(2076)	41	8	176	4	65 36 178 34 65 61 65 34 0
					A $ = " A = A "
(0829)	2F	08	50	C3	80 00
(2089)	47	8	80	195	128 0
					END
(082F)					
(2095)	0	0			

Bild 1. So legt der Interpreter Basic-Zeilen im Speicher ab.

```
  PC  SR AC XR YR SP  NV-BDIZC
;C00B B0 C2 AF 00 F6  10110000
.M0800
:0800 00 0F 08 0A 00 99 20 22   ......"
:0808 50 52 4F 42 45 22 00 1C   PROBE"..
:0810 08 2C 01 8F 20 43 20 2D   .,.. C -
:0818 20 36 34 00 29 08 B0 04    64.)...
:0820 41 24 B2 22 41 3D 41 22   A$."A=A"
:0828 00 2F 08 50 C3 80 00 00   ./.P-...
.X
READY.
L

10 PRINT "PROBE"
300 REM C - 64
1200 A$="A=A"
50000 END
READY.
HARDCOPY
```

Bild 2. Das kleine Basic-Programm mit SMON betrachtet.

Mehr Verständnis für den NEW-Befehl

Im Handbuch steht, daß NEW das Programm im Speicher löscht. Das ist nur bedingt richtig, denn der NEW-Befehl löscht nicht das ganze Programm, sondern schreibt nur zwei Null-Codes in die Speicherstellen 2049 ($801) und 2050 ($802). Sie können das mit einem Monitor, zum Beispiel dem SMON, überprüfen. Außerdem werden die Zeiger für Variablenanfang, Feldanfang und Feldende auf die Adresse 2051 gesetzt.

Zum Beweis dieser Aussage sollten Sie einmal das Beispielprogramm eintippen und die folgenden Direktbefehle eingeben.

```
NEW
POKE 2049,15
POKE 2050,8   Linkpointer
POKE 45,49
POKE 46,8     Variablen-Anfang
POKE 47,49
POKE 48, 8    Feld-Anfang
POKE 49,49
POKE 50,8     Feld-Ende
LIST
```

Die Werte für den ersten Linkpointer und die Variablenzeiger werden mit diesen acht POKE-Befehlen wieder hergestellt. Sie gelten nur für dieses Beispielprogramm. Geben Sie nach NEW eine neue Basic-Zeile ein, dann kann das gelöschte Programm nicht mehr gerettet werden.

Wenn Sie nun am Ende dieses Satzes drei Nullen finden, dann wissen Sie
a) der Satz ist zu Ende und
b) der Artikel ist zu Ende.
000

(J.Effenberg/hm)

Befehl	Token DEZ	Token HEX	Befehl	Token DEZ	Token HEX	Befehl	Token DEZ	Token HEX
END	128	80	CONT	154	9A	SGN	180	B4
FOR	129	81	LIST	155	9B	INT	181	B5
NEXT	130	82	CLR	156	9C	ABS	182	B6
DATA	131	83	CMD	157	9D	USR	183	B7
INPUT #	132	84	SYS	158	9E	FRE	184	B8
INPUT	133	85	OPEN	159	9F	POS	185	B9
DIM	134	86	CLOSE	160	A0	SQR	186	BA
READ	135	87	GET	161	A1	RND	187	BB
LET	136	88	NEW	162	A2	LOG	188	BC
GOTO	137	89	TAB	163	A3	EXP	189	BD
RUN	138	8A	TO	164	A4	COS	190	BE
IF	139	8B	FN	165	A5	SIN	191	BF
REST.	140	8C	SPC	166	A6	TAN	192	C0
GOSUB	141	8D	THEN	167	A7	ATN	193	C1
RETURN	142	8E	NOT	168	A8	PEEK	194	C2
REM	143	8F	STEP	169	A9	LEN	195	C3
STOP	144	90	+	170	AA	STR$	196	C4
ON	145	91	—	171	AB	VAL	197	C5
WAIT	146	92	*	172	AC	ASC	198	C6
LOAD	147	93	/	173	AD	CHR$	199	C7
SAVE	148	94	↑	174	AE	LEFT$	200	C8
VERIFY	149	95	AND	175	AF	RIGHT$	201	C9
DEF	150	96	OR	176	B0	MID$	202	CA
POKE	151	97	>	177	B1	GO	203	CB
PRINT #	152	98	=	178	B2			
PRINT	153	99	<	179	B3			

Bild 3. Basic-Befehle und deren Token

Als die Bilder laufen lernten ...

Zaubern Sie Bewegung auf Ihren Bildschirm! Mit diesem Programm können Sie bequem Blockgrafik und Text in frei definierbaren Bildschirmbereichen scrollen. Und das in vier Richtungen.

Mit einem einzigen SYS-Befehl wird sowohl der Zeichen- als auch der Farbcode gescrollt. Dabei können Sie die Größe und Lage des Scrollbereiches frei wählen. Wenn Zeichen aus einem Bereich hinausgeschoben werden, tauchen Sie an der gegenüberliegenden Seite wieder auf.

Mit SYS 50550,r,za,ze,s,l wird das Maschinenprogramm aufgerufen. Dabei werden Parameter für Richtung (r), Zeilenanfang (za), Zeilenende (ze), Spalte (s) und Zeilenlänge (l) übergeben. Die Parameter müssen innerhalb der Grenzen von Bild 1 liegen. Angaben außerhalb dieser Grenzen fängt das Programm ab und gibt die entsprechende Fehlermeldung aus.

Listing 2 ist ein Demo-Programm, das Ihnen die verblüffende Wirkung des Scrollens in vier Richtungen zeigt.

(J. Effenberg/hm)

Richtung	r	L,R,H,T
Zeilenanfang	za	1 ... 25
Zeilenende	ze	1 ... 25, za<ze
Spalte	s	1 ... 40
Zeilenlänge	l	1 ... 40 s+l<41
Syntax		SYS50550,r,za,ze,s,l

Bild 1. Die Parametergrenzen

```
PROGRAMM : ROLLING              C576 C815
-----------------------------------------

C576 : 20 FD AE B1 7A C9 4C D0   3F
C57E : 07 E9 4C 8D A8 02 F0 22   E1
C586 : C9 52 D0 07 E9 51 8D A8   3E
C58E : 02 D0 17 C9 48 D0 07 E9   F2
C596 : 46 8D A8 02 D0 0C C9 54   4A
C59E : F0 03 4C 16 C7 E9 51 8D   12
C5A6 : A8 02 E6 7A D0 02 E6 7B   08
C5AE : 20 F1 B7 CA 30 09 E0 19   0F
C5B6 : B0 05 8E A9 02 50 03 4C   09
C5BE : 20 C7 20 F1 B7 CA 30 F7   8B
C5C6 : E0 19 B0 F3 8E AA 02 EC   FD
C5CE : A9 02 F0 05 B0 0D 4C 2A   4E
C5D6 : C7 AD A8 02 C9 02 90 03   D3
C5DE : 4C 52 C7 20 F1 B7 CA 30   B2
C5E6 : 04 E0 28 90 03 4C 34 C7   69
```

Listing 1. »Als die Bilder laufen lernten...«. Das Programm muß mit dem MSE auf Seite 68 eingegeben werden

```
C5EE : 8E AB 02 20 F1 B7 8A F0  BF
C5F6 : 0D E0 29 B0 09 6D AB 02  82
C5FE : C9 29 B0 05 90 06 4C 3E  0F
C606 : C7 4C 48 C7 CA 8E AC 02  D6
C60E : AD A8 02 F0 11 C9 01 F0  F3
C616 : 0A C9 02 F0 03 4C 83 C6  D2
C61E : 4C D4 C6 4C 56 C6 4C 27  2B
C626 : C6 AE A9 02 A0 00 20 FE  76
C62E : C6 B1 D1 48 B1 F3 48 C8  B8
C636 : B1 D1 48 B1 F3 88 91 F3  CA
C63E : 68 91 D1 C8 CC AC 02 D0  D8
C646 : EE 68 91 F3 68 91 D1 EC  7F
C64E : AA 02 B0 03 E8 D0 D5 60  B3
C656 : AE A9 02 AC AC 02 20 FE  48
C65E : C6 B1 D1 48 B1 F3 48 88  67
C666 : B1 D1 48 B1 F3 C8 91 F3  FC
C66E : 68 91 D1 88 D0 F1 68 91  86
C676 : F3 68 91 D1 EC AA 02 B0  C9
C67E : 03 E8 D0 D7 60 78 A0 00  71
C686 : 84 FB AE AA 02 20 FE C6  B3
C68E : B1 D1 48 B1 F3 48 CC AC  7E
C696 : 02 F0 03 C8 D0 F2 A4 FB  19
C69E : CA 20 FE C6 B1 D1 48 B1  3F
C6A6 : F3 48 E8 20 FE C6 68 91  E6
C6AE : F3 68 91 D1 CC AC 02 F0  90
C6B6 : 03 C8 D0 E4 CA EC A9 02  AD
C6BE : D0 DC 20 FE C6 AC AC 02  6D
C6C6 : 68 91 F3 68 91 D1 98 F0  ED
C6CE : 03 88 50 F4 58 60 CE 89  9F
C6D6 : C6 EE BC C6 A9 CA A2 E8  68
C6DE : 8D A8 C6 8E 9E C6 8E BA  12
C6E6 : C6 20 83 C6 A9 CA A2 E8  C3
C6EE : 8E A8 C6 8D 9E C6 8D BA  FF
C6F6 : C6 EE 89 C6 CE BC C6 60  1D
C6FE : 20 F0 E9 AD AB 02 F0 0C  6D
C706 : A5 D1 18 6D AB 02 85 D1  CC
C70E : 90 02 E6 D2 20 24 EA 60  43
C716 : A9 7F A0 C7 85 FB 84 FC  E4
C71E : 50 4D A9 95 A0 C7 85 FB  88
C726 : 84 FC 50 43 A9 A8 A0 C7  97
C72E : 85 FB 84 FC 50 39 A9 C2  6C
C736 : A0 C7 85 FB 84 FC 50 2F  6A
C73E : A9 D6 A0 C7 85 FB 84 FC  B8
C746 : 50 25 A9 EA A0 C7 85 FB  47
C74E : 84 FC 50 1B A9 FF A0 C7  74
C756 : 85 FB 84 FC 50 11 20 D7  57
C75E : AA A5 39 A6 3A 85 14 86  2B
C766 : 15 20 13 A6 4C C9 A6 20  13
C76E : D7 AA 20 D7 AA A0 00 B1  B0
C776 : FB 30 E3 20 D2 FF C8 D0  78
C77E : F6 52 49 43 48 54 55 4E  71
C786 : 47 53 57 45 52 54 20 49  D0
C78E : 4C 4C 45 47 41 4C FF 5A  66
C796 : 45 49 4C 45 4E 57 45 52  95
C79E : 54 20 49 4C 4C 45 47 41  6D
C7A6 : 4C FF 5A 45 49 4C 45 20  7D
C7AE : 31 20 49 53 54 20 3E 20  2C
C7B6 : 41 4C 53 20 5A 45 49 4C  84
```

```
C7BE : 45 20 32 FF 53 50 41 4C  F5
C7C6 : 54 45 4E 57 45 52 54 20  B4
C7CE : 49 4C 4C 45 47 41 4C FF  A9
C7D6 : 4C 41 45 4E 47 45 4E 57  64
C7DE : 45 52 54 20 49 4C 4C 45  18
C7E6 : 47 41 4C FF 53 50 41 4C  36
C7EE : 54 45 20 2B 20 4C 41 45  46
C7F6 : 4E 47 45 20 3E 20 34 30  53
C7FE : FF 5A 45 49 4C 45 20 31  77
C806 : 20 49 53 54 20 3D 20 5A  4B
C80E : 45 49 4C 45 20 32 FF 88  58
```

Listing 1. »Als die Bilder laufen lernten ...«. Das Programm muß mit dem MSE eingegeben werden (Schluß).

```
1 POKE 53281,0:POKE 53280,0:PRINT"{GREEN}" <091>
10 GOSUB 1000                              <088>
40 FOR I=1 TO 100000                       <154>
50 SYS 50550,H,2,9,1,10                    <240>
60 SYS 50550,T,17,24,1,10                  <105>
62 SYS 50550,T,19,24,11,10                 <158>
63 SYS 50550,H,19,24,20,5                  <103>
65 SYS 50550,L,12,14,5,30                  <102>
80 SYS 50550,R,16,24,29,10                 <180>
90 SYS 50550,L,1,8,16,24                   <085>
100 A=A+1:IF A/2=INT(A/2)THEN PRINT"{UP}
   "TAB(15)"ATTENTION"                     <113>
110 IF A/3=INT(A/3)THEN PRINT"{UP}"TAB(15)"
   {RVSON}ATTENTION"                       <070>
999 NEXT                                   <108>
1000 PRINT"{CLR}"                          <091>
1001 PRINT"{4SPACE}NM{22SPACE}"            <100>
1010 PRINT"{3SPACE}N{2SPACE}M{5SPACE,RED,7SPACE,
   RVSON}£{RVOFF}H{6SPACE,GREEN}"          <168>
1020 PRINT"{2SPACE}N{4SPACE}M{4SPACE,RED,6SPACE,
   RVSON}£{SPACE,RVOFF}H{6SPACE,GREEN}"    <178>
1025 PRINT" N{6SPACE}M{3SPACE,RED,5SPACE,RVSON}
   £{2SPACE,RVOFF}LPPPPPP{GREEN}"          <234>
1026 PRINT" M{6SPACE}N{3SPACE,RED,5SPACE}*
   {RVSON,2SPACE,RVOFF}OYYYYYY{GREEN}"     <084>
1030 PRINT"{2SPACE}M{4SPACE}N{4SPACE,RED,6SPACE}
   *{RVSON,SPACE,RVOFF}H{6SPACE,GREEN}"    <243>
1040 PRINT"{3SPACE}M{2SPACE}N{5SPACE,RED,7SPACE}
   *H{6SPACE,GREEN}"                       <089>
1050 PRINT"{4SPACE}MN{22SPACE}"            <150>
1060 PRINT                                 <193>
1070 PRINT"{RVSON,39SPACE}"                <033>
1090 PRINT"{4SPACE,RVSON}...S Y S T E M{3SPACE}
   R U N N I N G"                          <197>
1100 PRINT"{RVSON,39SPACE}"                <063>
1170 PRINT                                 <047>
1180 PRINT                                 <057>
2000 PRINT"IIIIIIIIIF{18SPACE}@@{9SPACE}"  <195>
2005 PRINT"{GREY 3}OOOOOOOOOK{19SPACE}@@{8SPACE}
   "                                       <024>
2010 PRINT"{GREY 2}OOOOOOOOOK{20SPACE}@@{7SPACE}
   "                                       <026>
2015 PRINT"{PURPLE}OOOOOOOOOK{21SPACE}@@{6SPACE}
   "                                       <035>
2020 PRINT"{GREY 3}OOOOOOOOOK{5SPACE}POKE DOWN
   {8SPACE}@@{5SPACE}"                     <142>
2025 PRINT"{GREY 2}OOOOOOOOOK{23SPACE}@@{4SPACE}
   "                                       <041>
2030 PRINT"{YELLOW}OOOOOOOOOK{24SPACE}@@{3SPACE}
   "                                       <052>
2035 PRINT"{9SPACE}K{25SPACE}@@{2SPACE}"   <026>
2040 PRINT"{9SPACE}K{26SPACE}@@ "          <031>
3000 PRINT"{HOME,15DOWN}"                  <178>
10000 RETURN                               <197>
```

Listing 2. Dieses Demo-Programm zeigt Ihnen die verblüffende Wirkung des Scrollens in allen Richtungen

Besseres Monitorbild beim C 64

Mit dieser kleinen Bauanleitung läßt sich die Bildschirmwiedergabe des C 64 auf einem monochromen Monitor wesentlich verbessern.

Nachdem ich einen monochromen Monitor (zirka 300 Mark, 15 MHz, grün) an die Videobuchse (Luminanzsignal) meines C 64 angeschlossen hatte, konnte ich das Testergebnis der Ausgabe 12/84 nur bestätigen, »daß es nicht so einfach ist, dem Commodore 64 ein klares und kontrastreiches Bild zu entlocken«.

Da bekanntermaßen ein Signal (auf dem Monitor) nur so gut abgebildet werden kann, wie es von der Quelle (C 64) ausgesendet wird, untersuchte ich mit Hilfe eines Oszilloskops die Signalverläufe innerhalb der Modulator- beziehungsweise Videoschaltung. Dabei stellte ich fest, daß der vom VIC II-Chip unter anderem bereitgestellte Zeileninhalt in der Videostufe durch einen Tiefpaß auf wenige MHz begrenzt wird. Das hat zur Folge, daß die Ränder der auf dem Monitor dargestellten Zeichen oder Punkte nur unscharf abgebildet, und die horizontalen Bildanteile (insbesondere bei Grafik) stark überbetont werden.

Überraschenderweise fand ich bei weiterem Durchmessen der Videostufe einen Meßpunkt (Punkt »A« in der Skizze), an dem das vollständige Videosignal (FBAS) mit **nicht** begrenzter Bandbreite vorliegt, und der mit dem niederohmigen Eingang (75 Ω) des Monitors belastet werden darf.

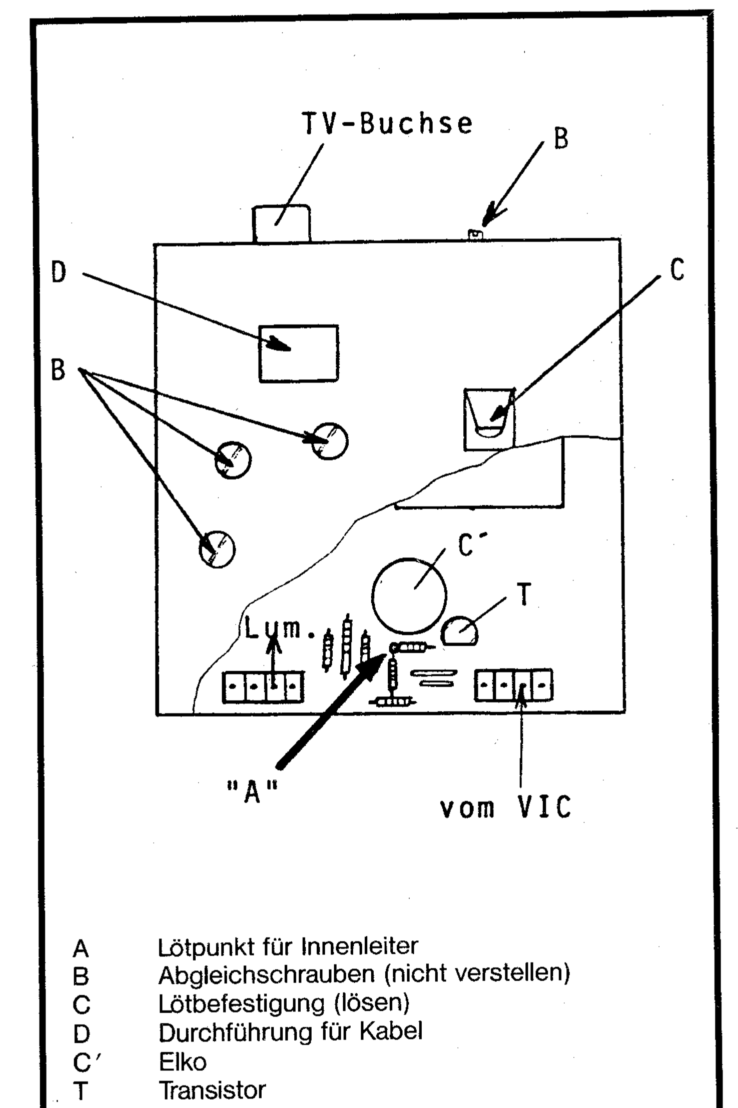
Unter der Voraussetzung, daß die vier vorhandenen Abgleicheinrichtungen nicht verstellt werden (Punkte »B« in der Skizze) und die Garantieansprüche unter Umständen durch diesen Eingriff verfallen, kann das Monitorbild des Commodore 64 folgendermaßen verbessert werden:

- Öffnen des C 64-Gehäuses mit den auf der Computerrückseite vorhandenen drei Schrauben.
- Entfernen der Abschirmpappe und Abziehen der Tastatur- und LED-Leitungen.
- Lösen des Lötpunktes »C« bei gleichzeitigem Abheben des Modulatorgehäusedeckels.
- Anlöten des Innenleiters eines handelsüblichen, abgeschirmten Kabels an der mit »A« gekennzeichneten Stelle. Diese befindet sich zur Kontrolle zwischen einem 120 Ω und 180 Ω Widerstand.
- Anlöten der Kabelabschirmung an der Gehäuseinnenwand.
- Durchführen des Koaxialkabels durch die Öffnung »D« im Gehäusedeckel sowie durch eine der Öffnungen auf der Rechnerrückseite.
- Anlöten eines handelsüblichen Cinch-Steckers am Ende des zirka 1 m bis 2 m langen Koaxkabels.
- Schließen und Verlöten des Modulatordeckels, Aufstecken der abgezogenen Leitungen und Zusammenbau des C 64.

Um ein möglichst kontrastreiches Bild zu erzielen, empfiehlt sich bei monochromen Monitoren ein schwarzer Hintergrund und weiße Zeichen.

Es sei noch einmal darauf hingewiesen, daß die Versuche mit einem monochromen Monitor durchgeführt wurden. Ob sich die Qualität von Farbmonitorbildern auf die gleiche Weise verbessern läßt, müssen am besten weitere Experimente zeigen.

(Dipl.-Ing. Stephan Greitzke/aa)



Schematisierte, auszugsweise Darstellung der Modulatorplatine mit Gehäuse

Maschinen-programme auf Diskette speichern

Reine Maschinenprogramme haben gegenüber einem Basic-Lader zwei entscheidende Vorteile. Sie sind wesentlich kürzer und können direkt, ohne zeitraubendes POKEn, geladen werden. Dieses Programm soll Ihnen helfen, Maschinenprogramme auf Diskette zu speichern.

Ein einfacher Basic-Lader besteht aus einer READ/POKE-Schleife, in der DATAs in den Speicher geschrieben werden. Jeder Wert, der in dieser Schleife gePOKEt wird, ist ein Byte eines Maschinenprogramms, ein Maschinenbefehl oder ein Teil davon. Anstatt die DATAs in den Speicher zu POKEn, kann ein Basic-Lader dazu benutzt werden, das Maschinenprogramm auf Diskette abzuspeichern. Der Lader wird dazu nur etwas geändert.

Ein Maschinenprogrammfile auf Diskette enthält, vor den Maschinenbefehlen, in den ersten beiden Bytes, die Startadresse des Programms. Der Computer erkennt daran, ab welcher Speicherstelle er das Programm laden soll.

Der Basic-Lader muß also ein Programmfile eröffnen, die Startadresse des Maschinenprogramms ins File schreiben und anschließend alle Werte aus den DATA-Zeilen. Wie das im einzelnen programmiert wird, können Sie am Listing des Beispielprogramms sehen, das Sie ohne weiteres an jedes Ladeprogramm anpassen können.

Der wichtigste Befehl in diesem Beipiel steht in Zeile 50130:
OPEN 1,8,1, "Programmname"

Er öffnet eine Datei. Allerdings keine sequentielle, sondern eine Programmdatei. Die Sekundäradresse 1 sagt der Floppy 1541, daß nun ein Programm (NAME.PRG) geschrieben wird. Häufig findet man für diesen Befehl die Syntax
OPEN 2,8,2, "Programmname,P, W"

bei der man die Sekundäradresse frei wählen darf. Hier muß jedoch nach dem Namen angegeben werden, daß eine Programmdatei (P) geöffnet werden soll, in die geschrieben wird (Write). Entsprechend kann bei sequentiellen Dateien optional »S,W« bei Schreib- oder »S,R« bei Lesezugriffen angegeben werden.

Nachdem in Zeile 50130 die Programmdatei angelegt wurde, muß die Startadresse des Maschinenprogrammes gespeichert werden. Die Startadresse wird dazu in den Zeilen 50170 und 50180 in Low- und High-Byte zerlegt und die zwei Byte in Zeile 50190 in das Programmfile geschrieben. Jetzt wird das eigentliche Programm gespeichert. Die numerischen Werte müssen dazu mit der CHR$-Funktion in die entsprechenden Strings übersetzt werden. Um die erforderliche Datendichte beim Schreiben des Files zu erreichen, muß jedem String ein Semikolon »;« folgen. Ohne Semikolon ist das Programm später nicht lauffähig. Sind alle Werte, die der Basic-Lader in den Speicher gePOKEt hätte, gespeichert, wird die Datei mit CLOSE geschlossen. Damit ist das Maschinenprogramm auf Diskette gespeichert und kann mit
LOAD "NAME",8,1

geladen und, wenn dies notwendig ist, mit dem entsprechenden SYS-Befehl gestartet werden.

Laden von Maschinenprogrammen in Basic

Sie können Maschinenroutinen mit Hilfe von Basic-Programmen laden, ohne sie zu zerstören. Dabei ist zu beachten, daß nach dem Laden das Basic-Programm von Neuem startet. Die Basic-Zeile
10 LOAD "KEIN ENDE",8,1

bewirkt deshalb eine Endlosschleife, die immer wieder das Maschinenprogramm »KEIN ENDE« lädt. Da angelegte Variablen dabei erhalten bleiben, kann mit
10 IF A=0 THEN A=1:LOAD "KEIN ENDE",8,1
ein mehrmaliges Laden verhindert werden, da beim Neustart A=1 ist.

(S. Wengler/hm)

Listing des Beispielprogramms

```
50000 REM   BASIC-LADER
50010 DATA  ..........................
50020 DATA  ..........................
50030 DATA  ..........................
50040 :
50050 REM   MASCHINENPROGRAMME AUS
50060 REM   DATALADERN
50070 :
50080 REM S  KOENNEN WEGGELASSEN WERDEN
50090 :
50100 REM   SA = STARTADRESSE
50110 REM   EA = ENDADRESSE
50120 :
50130 OPEN 1,8,1, "NAME"
50140 :
50150 REM  SA U. EA ZUORDNEN
50160 :
50170 SH = INT(SA/256)  :REM HIGH-BYTE
50180 SL = SA-SH*256    :REM  LOW-BYTE
50190 PRINT#1,CHR$(SL);CHR$(SH);
50200 :
50210 FOR I=SA TO EA
50220 READ WERT
50230 PRINT#1,CHR$(WERT);
50240 NEXT I
50250 :
50260 CLOSE 1
```

RAM-Floppy

Wer kennt das nicht: ein paar Veränderungen an einem Programm — eine Zeile rein, eine andere raus — und nichts geht mehr. Das lästige Neuladen des Originalprogramms von Diskette können Sie ab jetzt vergessen.

Ist ein Programm mal wieder zu Tode editiert, werden Sie nun nicht mehr von den langen Ladezeiten der 1541 in Ihrem Programmierdrang gebremst. Mit »RAM-Floppy« kann ein Programm bearbeitet werden, während man eine Kopie davon im RAM hat. In Sekundenschnell kann die Kopie in den Basic-Speicher gebracht oder mit der Originalversion vertrauscht werden. Ganz einfach durch Eingabe von »@V« oder »@T«. Numerische Variablen bleiben dabei erhalten. Die »RAM-Floppy« besitzt eine Speicherkapazität von maximal 25 KByte. Der Speicher beginnt ab Adresse 40960.

Ein Problem ergibt sich im Speicherbereich des Kernal- und Basic-ROMs. Ein POKE-Befehl schreibt ins RAM, während die PEEK-Funktion auf das ROM zugreift. Noch komplizierter sieht es beim Zeichen-ROM und den I/O-Bausteinen aus. Wie Sie vielleicht aus unserem Grafikkurs wissen, gibt es in diesen Bereichen drei Speicheretagen. Der Inhalt der Zelle 1 regelt den Zugriff des Computers auf die verschiedenen Speicherebenen. Werden die Bits 0 und 1 in Adresse 1 gelöscht, sieht der Computer nur noch das RAM. Basic- und Kernal-ROM sind verschwunden. Löschen Sie diese Bits deshalb nur durch ein Maschinenspracheprogramm, wenn vorher Ein- und Ausgaben gesperrt wurden. Dies wird durch Setzen des Interruptregisters erreicht.

Das Maschinenprogramm besteht aus drei Teilen. Im Bereich von 40704 bis 40768 erfolgt die Auswertung der Befehle von »RAM-Floppy« und der Aufruf der beiden Unterprogramme, die das Tauschen oder Verschieben der Basic-Programme erledigen.

Das abgedruckte Basic-Programm POKEt das Maschinenprogramm ab Adresse 40704. Mit SYS 40704 wird es initialisiert.

Wie bereits erwähnt, hat die »RAM-Floppy« eine Kapazität von 25 KByte. Das Programm im Basic-Speicher kann zwar 38 KByte lang sein, läßt sich dann allerdings nicht mehr vollständig verschieben oder vertauschen. Der Speicherbedarf sollte auch bei Programmen mit vielen Variablen nicht außer acht gelassen werden. Bei langen Programmen mit vielen Variablen kann es durchaus vorkommen, daß die Programme zwar getauscht, die Variablen allerdings nicht mehr übernommen werden können.

(Uwe Klatt/hm)

Listing zu »Ram-Floppy«

```
0 REM************************************       <035>
1 REM*            RAM-FLOPPY               *     <203>
2 REM************************************       <037>
3 REM*  UWE KLATT                          *     <087>
4 REM*  BILLERBECKER STR. 27               *     <221>
5 REM*  4939 STEINHEIM                     *     <103>
6 REM*  TEL. 05233/5672                    *     <252>
7 REM************************************       <042>
8 POKE 53280,0:POKE 53281,11                    <095>
9 POKE 646,8                                    <164>
10 PRINT"BITTE WARTEN"                          <048>
11 REM ****************************             <092>
12 REM *** DATAS  LESEN          ***            <123>
13 REM ****************************             <094>
14 FOR I=40704 TO 40768:READ A:POKE I,A:S=S+A
   :NEXT                                        <248>
15 IF S<>6567 THEN END                          <080>
16 FOR I=40784 TO 40849:READ A:POKE I,A:S=S+A
   :NEXT                                        <002>
17 IF S<>14392 THEN END                         <125>
18 FOR I=40853 TO 40902:READ A:POKE I,A:S=S+A
   :NEXT                                        <247>
19 IF S<>20412 THEN END                         <117>
20 REM ****************************             <101>
21 REM ***        MENUE          ***            <026>
22 REM ****************************             <103>
23 PRINT CHR$(147)                              <100>
24 PRINT"{RVSON,40SPACE}";                      <066>
25 PRINT"{RVSON,SPACE}RAM-FLOPPY 25.5 KBYTE
   {18SPACE}";                                  <115>
26 PRINT"{RVSON,40SPACE}"                       <009>
27 PRINT"'@V'{2SPACE}VERSCHIEBT PROGRAMM IN
   {2SPACE}RAM-FLOPPY"                          <174>
28 PRINT"'@T'{2SPACE}VERTAUSCHT PROGRAMM MIT
   RAM-FLOPPY"                                  <026>
29 REM ****************************             <110>
30 REM *** MC PROGRAMM STARTEN   ***            <191>
31 REM ****************************             <112>
32 SYS 40704                                    <121>
33 REM ****************************             <114>
34 REM *** DATAS FUER 1. MC TEIL ***            <105>
35 REM ****************************             <116>
36 DATA 169,159,133,56,133,52,169,0,133,55,169,
   21,141,8,3,169                               <239>
37 DATA 159,141,9,3,96,32,115,0,240,4,201,64,
   240,3,76,231                                 <011>
38 DATA 167,32,115,0,201,84,240,7,201,86,240,12,
   76,8,175,32                                  <116>
39 DATA 115,0,32,80,159,76,174,167,32,115,0,32,
   149,159,76,174                               <239>
40 DATA 167                                     <073>
41 REM ****************************             <122>
42 REM *** DATAS FUER 2. MC TEIL ***            <114>
43 REM ****************************             <124>
44 DATA 169,0,133,45,169,104,133,46,120,165,1,
   41,252,133,1,169                             <015>
45 DATA 0,133,98,133,100,141,0,160,169,160,133,
   101,169,8,133,99                             <066>
46 DATA 162,96,160,0,177,98,133,102,177,100,145,
   98,165,102,145,100                           <227>
47 DATA 200,208,241,230,99,230,101,202,208,232,
   165,1,9,3,133,1                              <253>
48 DATA 88,96                                   <190>
49 REM ****************************             <130>
50 REM *** DATAS FUER 3. MC TEIL ***            <123>
51 REM ****************************             <132>
52 DATA 120,165,1,41,252,133,1,169,0,133,98,133,
   100,141,0,160                                <206>
53 DATA 169,160,133,101,169,8,133,99,162,96,160,
   0,177,98,145,100                             <151>
54 DATA 200,208,249,230,99,230,101,202,208,240,
   165,1,9,3,133,1                              <011>
55 DATA 88,96                                   <197>
```

Hires-3 — eine Super-Grafikerweiterung zum Grafikkurs

Diese Erweiterung bietet wesentlich mehr als ein einfaches Hilfsprogramm. Sie stellt einen umfangreichen Befehlskatalog für die Erzeugung von Grafiken zur Verfügung. Und sie liefert außerdem etliche Programmierhilfen.

Es hat ein bißchen gedauert, doch nun will ich mein Versprechen einlösen: Dornröschen ihre Behäbigkeit mit einem grafikunterstützenden Maschinenprogramm zu nehmen. Zuvor noch zwei erfreuliche und eine weniger gute Nachricht: Leider ist das Grafik-Hilfsprogramm so lang, daß es Ihnen in zwei Teilen vorgelegt werden muß. Dann die beiden frohen Botschaften: Sie bekommen etwas viel besseres als nur eine kleine Grafikhilfe und die Grafikserie wird fortgesetzt.

Viele Briefe zeigten uns, daß Sie noch mehr wissen möchten. Zu speziellen Grafikthemen soll daher in lockerer Reihenfolge weiterhin Dornröschen aufgeputzt werden. Wenn sie also spezielle Grafikwünsche haben, dann schreiben Sie. Der C 64 kann viel mehr als Sie vielleicht glauben! Was Apple kann — behaupte ich mal ganz frech — das kann unser Computer auch — nur viel preiswerter.

Wieso bekommen Sie etwas viel besseres als ursprünglich vorgesehen? Das zunächst vorgesehene Grafik-Hilfsprogramm ist Ende 1983 fertiggestellt worden und hieß Hires-1. Es war der Auslöser für diese Serie. Ich hatte es entwickelt, um meine eigenen Grafikbedürfnisse (Punkte setzen, Linien ziehen etc.) auf schnelle Art mit SYS-Aufrufen zu erfüllen. Die Entwicklung blieb aber nicht stehen. Hires-2 holte sich bereits alle nötigen Werte zum Zeichnen mit den USR-Kommandos und die SYS-Befehle wurden überflüssig. Was Sie jetzt bekommen, das ist Hires-3. Aufrufe der einzelnen Routinen (es sind über 40) erfolgen mit neu geschaffenen Basic-Befehlsworten, die sowohl im Direktmodus als auch in Programmen verwendet werden. Kenner werden feststellen, daß Hires-3 sozusagen organisch gewachsen ist, wie es halt vielen großen Programmprojekten ergeht, die immer weiter verbessert werden: Nebenbei bemerkt: Momentan arbeite ich an Hires-4.

Hires-3 wurde mit dem Gedanken entworfen, das Schreiben von Grafikprogrammen einfacher zu gestalten. Deswegen sind nicht nur ausgesprochene Grafikbefehle dabei, sondern auch OLD, MERGE, RENUMBER, AUTONUMBER, PAUSE, UHR etc. Die Grafik-Unterstützung ist für alle Anwendungsbereiche gedacht, wobei der Schwerpunkt allerdings weniger auf »Spielen« und »künstlerische Anwendungen« liegt. Deswegen sind keine Joystick-, keine Lightpen- und keine Sprite-Befehle enthalten. Schließlich spielte es noch eine große Rolle, daß viel Speicherplatz für Basic-Programme und Daten frei bleiben sollte, ebenso wie der für Maschinensprache interessante Bereich von $C000 bis $D000.

Deswegen liegt das Programm Hires-3 ab $8000 (dez.32768) bis $89B5 (dez.35253) und von $9000 (dez.36864) bis $9DCB (dez.40395). Es fängt also dort an, wo auch Module ihren Platz haben. Der Bildschirm für hochauflösende Grafik liegt von $8C00 (dez.35840) bis $8FFF (dez.36863), der für den Normalbetrieb bleibt an der gewohnte Stelle (1024...). Die 8000 Byte lange Bit-Map kommt uns nicht mehr in die Quere: Sie ist unter dem Basic-ROM versteckt. Wir haben also trotz 44 neuer Befehle und Funktionen und einer stets präsenten hochauflösenden Grafik auf einem zweiten Bildschirm immer noch 30 KByte RAM für Basic frei und zusätzlich auch noch den Bereich $C000 bis $D000 für Maschinenprogramme.

Hires-3 wird mit dem MSE eingegeben (Seite 68). Nach der Eingabe des Programms empfiehlt es sich, das Ganze erstmal auf einem Massenspeicher (Kassette oder Diskette) sicher abzulegen.

Bevor Sie ein neues Basic-Programm schreiben, sichern Sie bitte noch HIRES-3 durch die in den vergangenen Grafik-Folgen vorgestellten Schutz-POKEs: POKE52,128:POKE56,128

Mit SYS36864 starten Sie Hires-3. Außer der READY-Meldung werden Sie erstmal nichts Neues sehen. Sie haben nämlich nichts anderes getan, als den Funktionstasten Befehle zuzuteilen. Deshalb fangen wir jetzt mit der Erklärung der Befehle an und zwar mit eben diesen Funktionstasten: Vorsicht! Nicht alles gleich ausprobieren! Sehen Sie vorher noch in der Tabelle 1 nach, welche Befehle erst in der nächsten Folge von Hires-3 eingebaut werden.

Tabelle 1. Von den im Artikel erwähnten Befehlen kommen die nachfolgenden erst im nächsten Teil von Hires-3

— F2	RENUMBER
—	DUMP
—	UHR

Wenn Sie die vorher benutzen, stürzt das Programm wahrscheinlich ab. Es erfolgt dann ein Programmsprung ins Leere — und das wollen wir unserem Computer doch nicht zumuten!

1. Hilfsfunktionen

1.1 Funktionstastenbelegung

Genau genommen schaltet man durch SYS 36864 nur die Belegung der Funktionstasten ein. Ausschalten erfolgt durch gleichzeitiges Drücken der RUN/STOP- und der RESTORE-Tasten. Nun zur Belegung im einzelnen:

F1 RUN

Startet ein im Basic-Speicher stehendes Programm sofort.

F2 RENUMBER-Befehl

Auf dem Bildschirm erscheint SYS33256. Man schreibt nun: SYS33256,Abz,Nbz,Sch.

Dabei bedeuten:

Abz = erste alte Basic-Zeile, von der an neu numeriert werden soll.

Nbz = erste neue Basic-Zeilennummer

Sch = Schrittweite

Dieser Befehl numeriert auch alle GOTO, GOSUB, IF...THEN..., RUN... etc. neu.

Beispiel:

SYS33256,1,10,5 numeriert ein Programm ab der alten Zeilennummer 1, welche nun 10 heißt, in 5er-Schritten neu durch.

F3 Aktivieren der neuen Basic-Befehle

Auf dem Bildschirm erscheint SYS37498 und READY. Jetzt sind alle neuen Basic-Befehlsworte verfügbar. Verwendet man eines der neuen Befehlsworte vorher, erzeugt das einen SYNTAX ERROR. Das Abschalten der neuen Befehle geschieht mittels »AUS«.

F4 OLD-Befehl

Auf dem Bildschirm erscheint SYS37227 und ein durch NEW oder einem RESET gelöschtes Basic-Programm wird wieder gelistet und funktionsfähig (vorausgesetzt, daß es nicht durch Überschreiben zerstört wurde). Störungen können auftreten, wenn noch kein Basic-Programm im Speicher enthalten ist. Nach der Verwendung dieses Befehls sollte überprüft werden, ob durch das vorangegangene Löschen die SchutzPOKEs 52 und 56 verändert worden sind.

F5 AUTONUMBER-Befehl

Auf dem Bildschirm erscheint SYS37018 und eine Zeilennummer. Diese Funktion setzt automatisch nach jedem RETURN eine neue Zeilennummer, die sich an die letzte im Programm vorhandene anschließt. Im Normalfall wird dabei in 10er-Schritten gearbeitet, eine andere Schrittweite wird durch POKE37169, Sch erzeugt (dabei ist Sch die Schrittweite). Die AUTONUMBER-Funktion wird durch Drücken von RETURN direkt nach einer Zeilennummer abgeschaltet.

F6 MERGE-Befehl

SYS37306 erscheint auf dem

Bildschirm und die Information: »***MERGE***, BEREIT ZUM KOPPELN! READY.« Das Basic-Programm läßt sich allerdings nicht Listen oder Starten. Denn die Zeiger sind nun auf das Ende des Programms gestellt. Der Computer wartet auf das Laden des anzuhängenden Programms, was durch eine normale Ladeoperation stattfindet. Man drückt nun erneut F6 und auf dem Bildschirm erscheint: SYS37306 und der Text: »***MERGE*** PROGRAMME GEKOPPELT! READY«. Wenn man nun listet, zeigt es sich, daß beide Programme aneinandergehängt sind. Man sollte darauf achten, daß das anzuhängende Programm höhere Zeilennummern aufweist als das erste, denn sonst kommt es bei GOTO oder GOSUB-Sprüngen zu Fehlern.

F7 Startet Maschinenprogramme, die bei $C000 (dezimal 49152) liegen. Auf dem Bildschirm erscheint SYS49152.

F8 Startet Maschinenprogramme, die bei $7000 (dez. 28672) liegen. Auf dem Bildschirm erscheint SYS 28672.

Achtung! Diese beiden Tasten sollte man nur betätigen, wenn an den Ansprungadressen auch wirklich ein Programm startet, denn sonst stürzt der Computer unter Umständen ab. Sehr praktisch sind diese beiden Funktionstasten, wenn an den Startadressen andere Sprachen verfügbar sind, zum Beispiel bei $C000 ein Maschinensprache-Monitor liegt oder ähnliches. Auch zum schnellen Starten von unfertigen Assembler-Programmen, die an einer der beiden Stellen anfangen. Für solche Tests sind F7 und F8 gut einzusetzen.

1.2 Hilfsfunktionen als neue Basic-Befehle

Das sind fünf Befehle: PAU, DEEK, AUS, DUMP und UHR

Aus: Schaltet die Befehlserweitungen aus. Das beschleunigt den Ablauf eines Basic-Programms. Der Interpreter muß nun nicht mehr den Umweg über die hier vorgestellten Befehlserweiterungen laufen. Der Zeitunterschied ist allerdings minimal und nur bei zeitkritischen Abläufen interessant.

DEEK: Ist eine modifizierte PEEK-Funktion, die das RAM unter den ROM-Bausteinen ausliest und in Speicherstelle 2 ablegt.
Syntax: DEEK, Speicherstelle
Beispiel:
DEEK, 53272:PRINTPEEK(2)
ergibt 255, den Inhalt des RAM unter dem ROM, wogegen PRINTPEEK(53272) den ROM-Inhalt 21 ergibt.

PAU: Pausen-Befehl. PAU,10 erzeugt eine Pause von 10 Sekunden.

DUMP: Gibt alle definierten Variablen mit ihren aktuellen Werten aus. Arrays werden nicht berücksichtigt.

UHR: Zeigt in der rechten oberen Bildschirmecke eine Uhr an.
Stellen: UHR, »hhmmss«, Zeichenfarbe
Ausschalten: UHR
Wieder einschalten: UHR
Vor der Verwendung des Hochauflösungs-Modus sollte die UHR ausgeschaltet werden, ebenso vor dem Aufruf des Hardcopy-Befehls.

2. Grafik-Befehle

2.1 Einrichten der Grafik

HFL,Zf,Hf: Der Befehl richtet den Hochauflösungs-Grafikmodus ein, setzt die Zeichenfarbe (Zf) und die Hintergrundfarbe (Hf) und löscht ein eventuell vorhandenes Hochauflösungsbild aus. Dabei liegt der Bildschirm von $8C00 bis $8FE7 (dezimal 35840 bis 36839) und die Bit-Map unter dem Basic-ROM ($A000-$C000).

HAN: Schaltet nur den Hochauflösungs-Modus an.

FAR,Zf,Hf: Setzt nur die Farben im Hochauflösungsbild.

LOE: Löscht nur das Hochauflösungsbild.

HOF: Schaltet des Hochauflösungsbild aus und richtet den Normalmodus wieder ein.

2.2 Zeichnen im Bildschirmsystem

(dabei x von 0 bis 319, y von 0 bis 199). Erinnern Sie sich bitte an das Bildschirmkoordinatensystem. Alle x- und y-Koordinaten, die unter 2.2 und 2.3 benutzt werden, beziehen sich auf dieses System (Bild 1).

PKT,x,y: Zeichnet an der durch x und y angegebenen Stelle des Bildschirmes einen Punkt. Die Bedeutung der Bezeichnungen folgen aus Bild 2. Wegen der Eigenart des Bildschirmsystems ist darauf zu achten, daß x und y nie kleiner als 0, x nie größer als 319 und y nie größer als 199 werden. Koordinaten-Eingaben, die größer als 319 (beziehungsweise 199) sind, führen lediglich dazu, daß kein Punkt gezeichnet wird. Eingaben kleiner als Null ergeben einen SYNTAX ERROR.

LIN,xa,ya,xb,yb
Zeichnet eine Linie vom Punkt A mit den Koordinaten xa, ya bis zum Punkt B mit den Koordinaten xb, yb (siehe Bild 3). Die Richtung der Linie ist beliebig. Die Bemerkung zur Größe der Koordinaten gilt hier entsprechend.

REC,xa,ya,xb,yb
Zeichnet ein Rechteck, das durch den linken oberen Punkt A(xa,ya) und den rechten unteren Punkt B(xb,yb) gekennzeichnet ist (siehe Bild 4). Für die Koordinatengrenzwerte gilt dasselbe wie beim Befehl PKT.

BLO,xa,ya,xb,yb
Füllt ein Rechteck der angegebenen Maße (siehe REC für die Bezeichnungen) mit der Zeichenfarbe aus.

CIR,xm,ym,rx,ry,w
Zeichnet eine Ellipse oder einen

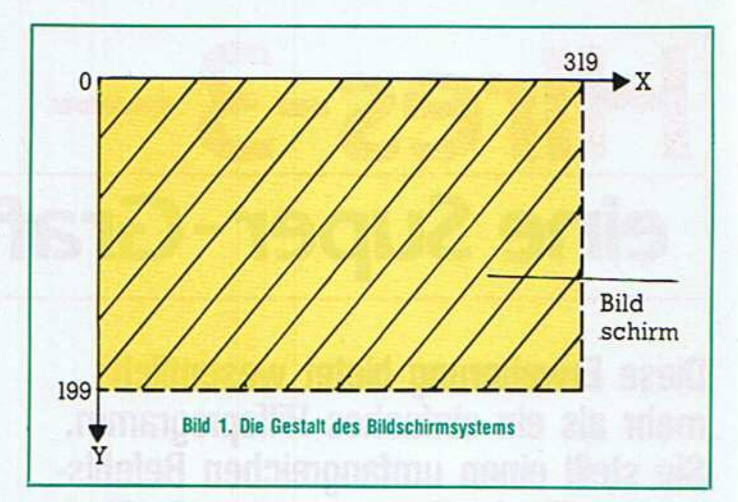


Bild 1. Die Gestalt des Bildschirmsystems

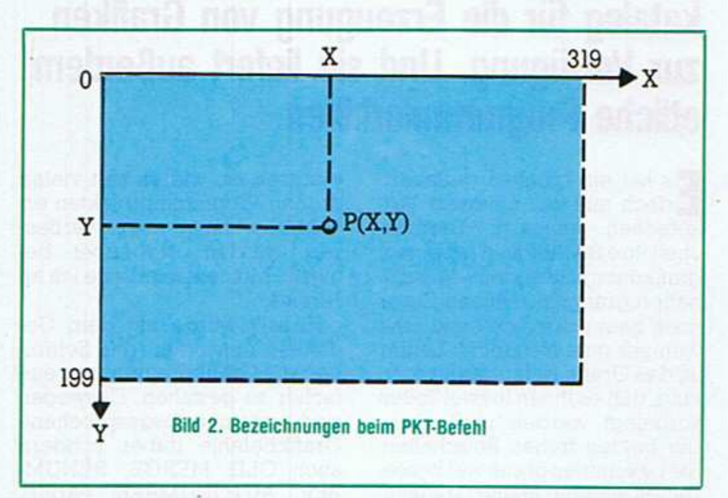


Bild 2. Bezeichnungen beim PKT-Befehl

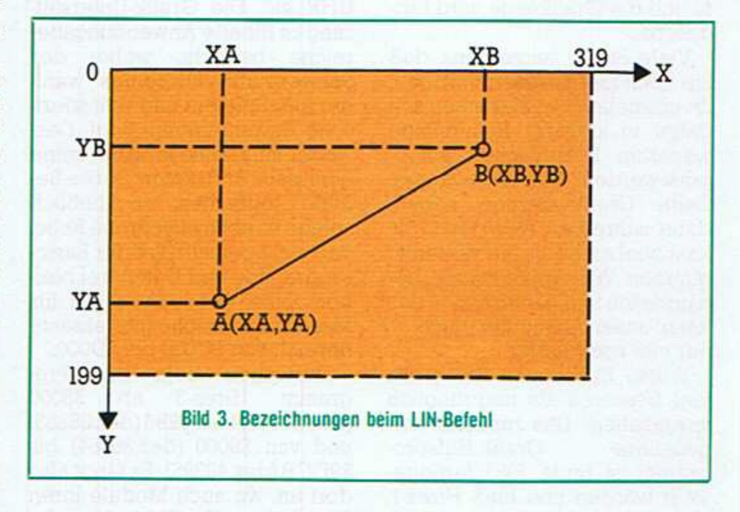


Bild 3. Bezeichnungen beim LIN-Befehl

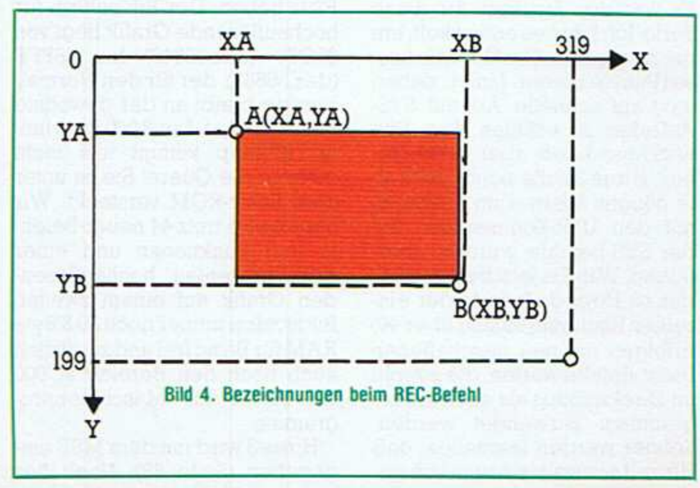


Bild 4. Bezeichnungen beim REC-Befehl

Kreis (Sonderfall der Ellipse mit rx = ry) mit den folgenden Merkmalen: xm,ym = Koordinaten des Mittelpunktes M, rx,ry = Halbmesser in x- beziehungsweise y-Richtung.
w = Zeichenwinkel (Bogenmaß).
Zur Erläuterung der einzelnen Bezeichnungen dient Bild 5. Außer negativen xm und ym-Koordinaten sind alle Eingaben zulässig.

RAD,xm,ym,rx,ry,w
Zeichnet in die Ellipse einen Radius ein. Die Bezeichnungen sind dieselben wie für den CIR-Befehl. w zeigt hier aber an, an welche Stelle im Ellipsenbogen der Radius gezeichnet werden soll (siehe Bild 6).

2.3 Löschen im Bildschirmsystem

LPK,x,y: Löscht den angegebenen Punkt x,y.
LLN,xa,ya,xb,yb: Löscht die Linie von xa,ya bis xb,yb
LRE,xa,ya,xb,yb: Löscht das Rechteck (Bezeichnungen wie bei REC)
LBK,xa,ya,xb,yb: Löscht das ausgefüllte Rechteck.
LKR,xm,ym,rx,ry,w: Löscht die Ellipse (Bezeichnungen wie bei CIR)
LRA,xm,ym,rx,ry,w: Löscht den Ellipsenradius

3. Abspeichern/Laden von Hochauflösungsbildern

HIS, "Name",gn,sa
SAVEn des Hochauflösungsbildes mit »Name« auf dem Gerät mit der Gerätenummer gn. Es gelten die gleichen Regeln wie beim normalen SAVE-Vorgang.
HIL, "Name",gn,sa
Laden eines Hochauflösungsbildes vom Speichermedium mit der Gerätenummer gn. Es gelten die gleichen Regeln wie für den normalen Ladevorgang.

Diese neuen Befehle sollen für diese Folge ausreichen. In der nächsten Ausgabe werden die anderen Befehle beschrieben, die HIRES-3 von allen mir bekannten Grafik-Software-Paketen unterscheidet.

Zur Praxis: Bedenken Sie bitte beim Ausprobieren dieser neuen Befehle, daß Sie alle auch im Direktmodus verwenden können. Das ist — gerade beim Testen — mitunter ganz bequem. Allerdings sind die Befehlsworte auf dem Hochauflösungsbildschirm nur als farbige Quadrate zu erkennen (Warum? Siehe vorangegangene Grafik-Folgen). Wenn Sie mal das Gefühl haben, Sie hätten sich beim Eintippen eines Befehls vergaloppiert, dann löschen Sie auch im Hochauflösungs-Modus den Bildschirm einfach mit SHIFT/CLR-HOME. Sie werden dann immer noch das Hochauflösungsbild sehen (die Bit-Map wird ja nur durch LOE gelöscht). Mit dem FAR-Befehl bringen Sie wieder neue Farbe ins Bild. Sollte ein Programm im Hochauflösungs-Modus auf einen Fehler laufen und aussteigen, dann kommen Sie mit HOF leicht wieder in den Normalmodus zurück.

Außer Hires-3 ist dieser Folge auch noch ein kleines Programm angefügt, mit dem Sie alle bisher vorgestellten Grafik-Befehle auf ihre Funktionsfähigkeit testen können. Nun noch ein Wort an die Spezialisten: In der nächsten Folge wird eine Aufstellung aller benötigten Zeropage-Plätze und der verwendeten Sub-Routinen erscheinen. Zur Selbstkritik: Einige Befehle fehlen, nämlich etwa zum Ausfüllen von umrandeten Flächen, dann eine Möglichkeit, Texte ins Hochauflösungsbild zu schreiben und anderes. Außerdem ist in puncto Geschwindigkeit noch lange nicht das Optimum erreicht... aber irgend etwas muß für Hires-4 auch noch zu tun bleiben.

(Heimo Ponnath/gk)

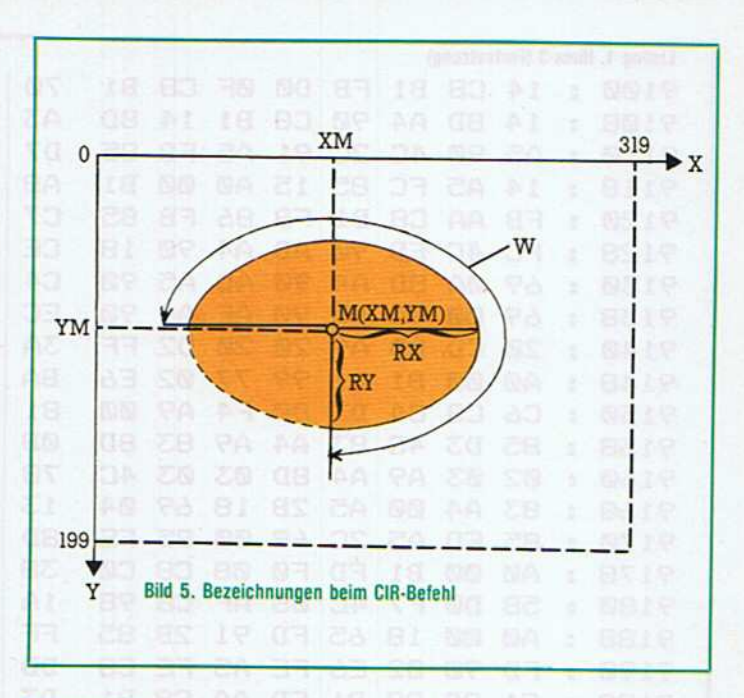


Bild 5. Bezeichnungen beim CIR-Befehl

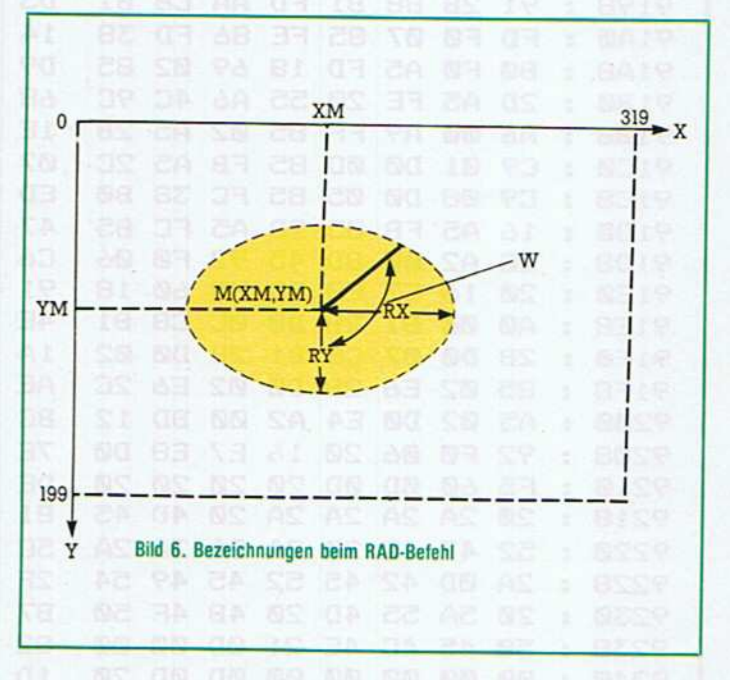


Bild 6. Bezeichnungen beim RAD-Befehl

Listing 1: Das Grafik-Programm Hires-3 muß mit dem MSE (siehe Seite 68) eingegeben werden

```
PROGRAMM : HIRES3.1.*        9000 9DCB
--------------------------------------
9000 : A9 0B A0 90 8D 8F 02 8C  DF
9008 : 90 02 60 A2 06 E4 CB F0  9E
9010 : 08 CA E0 02 D0 F7 4C 48  84
9018 : EB E4 C5 F0 F9 86 C5 AD  4B
9020 : 8D 02 C9 01 D0 04 E8 E8  E3
9028 : E8 E8 D8 A9 00 E0 03 F0  E5
9030 : 08 18 69 09 CA E0 03 D0  21
9038 : F8 AA A0 00 C8 BD 51 90  8E
9040 : 99 76 02 C9 0D F0 05 E8  0C
9048 : C0 09 30 F0 84 C6 4C 42  EB
9050 : EB 53 59 53 34 39 31 35  E2
9058 : 32 0D 52 55 4E 0D 00 00  9D
9060 : 00 00 00 53 59 53 33 37  36
9068 : 34 39 38 0D 53 59 53 33  9C
9070 : 37 30 31 38 0D 53 59 53  8A
9078 : 32 38 36 37 32 0D 53 59  C6
9080 : 53 33 33 32 35 36 00 53  2C
9088 : 59 53 33 37 32 32 37 0D  EA
9090 : 53 59 53 33 37 33 30 36  05
9098 : 0D FF A9 00 8D A4 90 8D  6B
90A0 : A5 90 F0 04 DC 05 AD 11  19
90A8 : A6 2D 8E A6 90 A6 2E 8E  71
90B0 : A7 90 A9 BC 8D 02 03 A9  E9
90B8 : 90 8D 03 03 A6 2D EC A6  05
90C0 : 90 D0 14 A6 2E EC A7 90  9C
90C8 : D0 0D AE A4 90 D0 05 AE  60
90D0 : A5 90 F0 03 4C 5D 91 A6  9D
90D8 : 2D 8E A6 90 A6 2E 8E A7  6D
90E0 : 90 A6 2D CA CA E4 2B D0  8A
90E8 : 06 A6 2E E4 2C F0 3D A6  F6
90F0 : 2B 86 FB 86 14 A6 2C 86  62
90F8 : FC 86 15 A0 00 B1 FB D0  B0
```

Listing 1. Hires-3 (Fortsetzung)

```
9100 : 14 C8 B1 FB D0 0F C8 B1 70
9108 : 14 8D A4 90 C8 B1 14 8D A3
9110 : A5 90 4C 2C 91 A5 FB 85 D7
9118 : 14 A5 FC 85 15 A0 00 B1 A8
9120 : FB AA C8 B1 FB 86 FB 85 C7
9128 : FC 4C FB 90 AD A4 90 18 CE
9130 : 69 0A 8D A4 90 AD A5 90 C4
9138 : 69 00 8D A5 90 AE A4 90 EC
9140 : 20 CD BD A9 20 20 D2 FF 3A
9148 : A0 00 B1 D1 99 77 02 E6 BA
9150 : C6 C8 C4 D3 D0 F4 A9 00 81
9158 : 85 D3 4C 83 A4 A9 83 8D 0B
9160 : 02 03 A9 A4 8D 03 03 4C 78
9168 : 83 A4 00 A5 2B 18 69 04 13
9170 : 85 FD A5 2C 69 00 85 FE 8D
9178 : A0 00 B1 FD F0 08 C8 C0 38
9180 : 58 D0 F7 4C 08 AF C8 98 1A
9188 : A0 00 18 65 FD 91 2B 85 FF
9190 : FD 90 02 E6 FE A5 FE C8 DD
9198 : 91 2B 88 B1 FD AA C8 B1 D3
91A0 : FD F0 07 85 FE 86 FD 38 14
91A8 : B0 F0 A5 FD 18 69 02 85 D9
91B0 : 2D A5 FE 20 55 A6 4C 9C 68
91B8 : A6 00 A9 FF 85 02 A5 2B 1E
91C0 : C9 01 D0 0D 85 FB A5 2C 07
91C8 : C9 08 D0 05 85 FC 38 B0 ED
91D0 : 16 A5 FB 85 2B A5 FC 85 47
91D8 : 2C A2 00 BD 45 92 F0 06 C6
91E0 : 20 16 E7 E8 D0 F5 60 18 91
91E8 : A0 00 B1 2B D0 0C C8 B1 4E
91F0 : 2B D0 07 C8 B1 2B D0 02 1A
91F8 : 85 02 E6 2B D0 02 E6 2C AE
9200 : A5 02 D0 E4 A2 00 BD 12 BC
9208 : 92 F0 06 20 16 E7 E8 D0 7E
9210 : F5 60 0D 0D 20 20 20 20 DE
9218 : 20 2A 2A 2A 2A 20 4D 45 81
9220 : 52 47 45 20 2A 2A 2A 2A 5C
9228 : 2A 0D 42 45 52 45 49 54 2F
9230 : 20 5A 55 4D 20 4B 4F 50 B7
9238 : 50 45 4C 4E 21 0D 00 00 82
9240 : 00 00 00 00 00 0D 0D 20 1D
9248 : 20 20 20 20 2A 2A 2A 2A 75
9250 : 20 4D 45 52 47 45 20 2A 26
9258 : 2A 2A 2A 2A 0D 50 52 4F A2
9260 : 47 52 41 4D 4D 45 20 47 D8
9268 : 45 4B 4F 50 50 45 4C 54 3A
9270 : 21 0D 00 00 00 00 00 00 18
9278 : 00 00 A9 92 8D 08 03 A9 AD
9280 : 92 8D 09 03 60 A9 E4 8D 7E
9288 : 08 03 A9 A7 8D 09 03 60 5F
9290 : 28 5F 20 73 00 90 1E C9 6F
9298 : 60 B0 1A C9 41 90 15 8D 18
92A0 : 91 92 A2 00 8E 90 92 A0 1C
92A8 : 00 EE 90 92 BD 40 93 D0 63
92B0 : 07 AD 91 92 38 4C E7 A7 19
92B8 : D1 7A D0 28 C8 E8 BD 40 4B
92C0 : 93 D0 F5 18 98 65 7A 85 E5
92C8 : 7A 90 02 E6 7B AD 90 92 74
92D0 : 0A AA BD EE 92 8D DF 92 B7
92D8 : BD EF 92 8D E0 92 20 BC 80
92E0 : 94 4C AE A7 E8 BD 40 93 DF
92E8 : D0 FA E8 4C A7 92 E7 A7 F7
92F0 : B0 94 B3 94 B6 94 B9 94 8A
92F8 : BC 94 74 9A E7 93 BF 94 B2
9300 : C2 94 C5 94 C8 94 CB 94 9A
9308 : CE 94 D1 94 D4 94 D7 94 A2
9310 : DA 94 DD 94 E0 94 E3 94 AA
9318 : E6 94 E9 94 EC 94 EF 94 B2
9320 : F2 94 F5 94 F8 94 FB 94 B9
9328 : FE 94 01 95 04 95 07 95 98
9330 : 0A 95 0D 95 85 92 DA 83 5A
9338 : 9D 84 B8 85 4F 87 E7 A7 16
9340 : 48 46 4C 00 48 41 4E 00 86
9348 : 46 41 52 00 4C 4F 45 00 18
9350 : 48 4F 46 00 50 41 55 00 36
9358 : 44 45 45 4B 00 50 4B 54 52
9360 : 00 4C 49 4E 00 52 45 43 D1
9368 : 00 42 4C 4F 00 43 49 52 6A
9370 : 00 52 41 44 00 48 49 53 80
9378 : 00 48 49 4C 00 4C 50 4B B2
9380 : 00 4C 4C 4E 00 4C 52 45 B9
9388 : 00 4C 42 4B 00 4C 4B 52 DD
9390 : 00 4C 52 41 00 54 52 53 06
9398 : 00 54 50 4B 00 54 4C 4E B0
93A0 : 00 54 52 45 00 54 42 4B 4A
93A8 : 00 54 4B 52 00 54 52 41 5E
93B0 : 00 4C 54 50 00 4C 54 4C 42
93B8 : 00 4C 54 52 00 4C 54 42 76
93C0 : 00 4C 54 4B 00 4C 54 56 C5
93C8 : 00 41 55 53 00 44 55 4D 3B
93D0 : 50 00 55 48 52 00 46 55 68
93D8 : 4E 4B 54 00 4C 46 55 4E CA
93E0 : 4B 00 00 00 00 00 00 A5 77
93E8 : 15 48 A5 14 48 20 FD AE E8
93F0 : 20 8A AD 20 F7 B7 A5 01 9A
93F8 : 48 29 FC 78 85 01 A0 00 06
9400 : B1 14 A8 68 85 01 58 68 85
9408 : 85 14 68 85 15 84 02 60 A0
9410 : A9 00 85 57 85 58 A0 08 B3
9418 : 18 26 5B 90 0D 18 A5 57 03
9420 : 65 59 85 57 A5 58 65 5A E5
9428 : 85 58 88 F0 06 26 57 26 55
9430 : 58 90 E6 60 A2 00 86 8E F7
9438 : 86 8F A0 10 06 57 26 58 14
9440 : 26 8E 26 8F 38 A5 8E E5 E0
9448 : 59 AA A5 8F E5 5A 90 06 D1
9450 : 86 8E 85 8F E6 57 88 D0 5D
9458 : E3 A5 57 85 5C A5 58 85 F4
9460 : 5D 60 85 36 49 24 92 87 25
9468 : 04 AA AA AB 83 E0 00 00 21
9470 : 00 81 40 00 00 00 00 00 41
9478 : 05 81 C0 00 00 00 86 3C 01
9480 : 00 00 00 83 49 88 2B 74 5F
9488 : 7B 2E 4C 41 5D 83 49 0F 8B
9490 : DA A1 33 01 75 34 01 6A FA
9498 : 01 00 FF 0B B8 00 0C 01 B8
94A0 : 00 01 3E 01 C6 A0 00 35 AC
94A8 : 00 88 47 00 00 00 00 00 BE
94B0 : 4C 87 95 4C 5D 95 4C 22 A7
94B8 : 95 4C 46 95 4C 72 95 4C FF
```

Listing 1. Hires-3 (Fortsetzung)

```
94C0 : 90 95 4C 1C 96 4C D2 97 F8
94C8 : 4C 88 98 4C FA 98 4C E8 7F
94D0 : 99 4C C4 9A 4C E8 9A 4C 23
94D8 : F7 9A 4C FF 9A 4C 07 9B 8F
94E0 : 4C 0F 9B 4C 17 9B 4C 1F E2
94E8 : 9B 4C 27 9B 4C A6 9B 4C E8
94F0 : 0B 9C 4C EB 9C 4C 00 80 07
94F8 : 4C E0 80 4C 4C 81 4C B8 D1
9500 : 81 4C C0 81 4C C8 81 4C B1
9508 : D0 81 4C D8 81 4C E0 81 C8
9510 : 4C AC 94 4C AC 94 4C AC 5B
9518 : 94 4C AC 94 4C AC 94 4C A5
9520 : AC 94 20 9B B7 8A 0A 0A 9E
9528 : 0A 0A 85 02 20 9B B7 8A AC
9530 : 05 02 A0 00 99 00 8C 99 5D
9538 : FA 8C 99 F4 8D 99 EE 8E FC
9540 : C8 C0 FA D0 EF 60 A9 A0 2B
9548 : 85 FE A9 00 85 FD A8 91 C5
9550 : FD C8 D0 FB E6 FE A4 FE 5C
9558 : C0 C0 D0 F2 60 A9 95 8D CF
9560 : 00 DD A9 38 8D 18 D0 A9 F0
9568 : 8C 8D 88 02 A9 3B 8D 11 EA
9570 : D0 60 A9 04 8D 88 02 A9 D4
9578 : 15 8D 18 D0 A9 97 8D 00 01
9580 : DD A9 1B 8D 11 D0 60 20 04
9588 : 22 95 20 46 95 4C 5D 95 A2
9590 : A9 00 8D A8 94 20 FD AE 51
9598 : 20 EB B7 EA EA EA EA 18 DB
95A0 : E0 C8 B0 6B A5 15 F0 0A 59
95A8 : C9 01 D0 63 A5 14 C9 40 35
95B0 : B0 5D A5 14 29 07 A8 38 D9
95B8 : A9 00 6A 88 10 FC 4D A8 7C
95C0 : 94 48 A5 14 29 F8 85 14 FD
95C8 : 8A 29 07 18 65 14 85 14 E1
95D0 : A5 15 69 A0 85 15 8A 29 EC
95D8 : F8 85 59 A9 00 85 5A A9 07
95E0 : 28 85 5B 20 10 94 18 A5 F7
95E8 : 14 65 57 85 14 A5 15 65 C3
95F0 : 58 85 15 68 A0 00 85 02 81
95F8 : A5 01 48 29 FE 78 85 01 21
9600 : A5 02 2C A8 94 30 0C 11 E4
9608 : 14 91 14 68 85 01 58 EA 8F
9610 : EA EA 60 31 14 91 14 68 9D
9618 : 85 01 58 60 A9 00 8D A8 62
9620 : 94 20 FD AE 20 EB B7 8E 77
9628 : 94 94 A5 14 8D 92 94 A5 FD
9630 : 15 8D 93 94 20 FD AE 20 70
9638 : EB B7 8E 97 94 A6 14 8E 81
9640 : 95 94 A5 15 8D 96 94 AE 69
9648 : 95 94 AD 96 94 CD 93 94 95
9650 : 30 0B F0 02 B0 2B EC 92 BF
9658 : 94 F0 6A B0 24 AD 92 94 38
9660 : AE 95 94 8D 95 94 8E 92 0D
9668 : 94 AD 93 94 AE 96 94 8D 57
9670 : 96 94 8E 93 94 AD 94 94 99
9678 : AE 97 94 8D 97 94 8E 94 4A
9680 : 94 38 AD 95 94 ED 92 94 7A
9688 : 8D 98 94 AD 96 94 ED 93 29
9690 : 94 8D 99 94 A2 00 8E 9A 7D
9698 : 94 38 AD 97 94 ED 94 94 DB
96A0 : F0 4F EE 9A 94 8D 9B 94 94
96A8 : B0 0B 49 FF 69 01 CA 8E 17
96B0 : 9A 94 8D 9B 94 AD 99 94 B1
96B8 : D0 62 AD 98 94 CD 9B 94 87
96C0 : B0 5A 4C 7D 97 AD 97 94 CE
96C8 : CD 94 94 B0 09 AE 94 94 9C
96D0 : 8E 97 94 8D 94 94 AE 94 D2
96D8 : 94 8A 48 AD 92 94 85 14 85
96E0 : AD 93 94 85 15 20 9B 95 19
96E8 : 68 AA E8 EC 97 94 90 E9 B1
96F0 : 60 AD 92 94 85 14 AD 93 35
96F8 : 94 85 15 AE 97 94 20 9B 40
9700 : 95 EE 92 94 D0 03 EE 93 4B
9708 : 94 AD 96 94 CD 93 94 90 98
9710 : 0A D0 DE AD 95 94 CD 92 4A
9718 : 94 B0 D6 60 A9 00 8D 9C D0
9720 : 94 8D 9D 94 18 AD 9C 94 FF
9728 : 85 59 6D 92 94 85 14 AD 29
9730 : 9D 94 85 5A 6D 93 94 85 95
9738 : 15 AD 9B 94 85 5B 20 10 71
9740 : 94 AD 98 94 85 59 AD 99 70
9748 : 94 85 5A 20 34 94 AD 94 01
9750 : 94 2C 9A 94 30 05 18 65 8A
9758 : 5C 90 03 38 E5 5C AA 20 F0
9760 : 9B 95 EE 9C 94 D0 03 EE CF
9768 : 9D 94 AD 99 94 CD 9D 94 45
9770 : 90 0A D0 B0 AD 98 94 CD DD
9778 : 9C 94 B0 A8 60 A9 00 8D 0E
9780 : 9E 94 85 5B AD 94 94 2C 5F
9788 : 9A 94 30 06 18 6D 9E 94 CA
9790 : 90 04 38 ED 9E 94 48 AD F9
9798 : 98 94 85 59 AD 99 94 85 0C
97A0 : 5A 20 10 94 AD 9B 94 85 B6
97A8 : 59 A9 00 85 5A 20 34 94 27
97B0 : 18 A5 5C 6D 92 94 85 14 6C
97B8 : A5 5D 6D 93 94 85 15 68 74
97C0 : AA 20 9B 95 EE 9E 94 AD A5
97C8 : 9E 94 CD 9B 94 F0 B3 90 58
97D0 : B1 60 A9 00 8D A8 94 20 CC
97D8 : FD AE 20 EB B7 8E A1 94 51
97E0 : A5 14 8D 9F 94 A5 15 8D CC
97E8 : A0 94 20 FD AE 20 EB B7 A5
97F0 : 8E A4 94 A6 14 8E A2 94 34
97F8 : A5 15 8D A3 94 AD 9F 94 5E
9800 : 8D 92 94 AD A0 94 8D 93 BD
9808 : 94 AD A1 94 8D 94 94 8D 59
9810 : 97 94 AD A2 94 8D 95 94 E6
9818 : AD A3 94 8D 96 94 20 47 8B
9820 : 96 AD A2 94 8D 92 94 8D A3
9828 : 95 94 AD A3 94 8D 93 94 14
9830 : 8D 96 94 AD A1 94 8D 94 01
9838 : 94 AD A4 94 8D 97 94 20 87
9840 : 47 96 AD 9F 94 8D 92 94 5B
9848 : AD A0 94 8D 93 94 AD A4 FA
9850 : 94 8D 94 94 8D 97 94 AD A6
9858 : A2 94 8D 95 94 AD A3 94 C9
9860 : 8D 96 94 20 47 96 AD 9F 81
9868 : 94 8D 92 94 8D 95 94 AD 2D
9870 : A0 94 8D 93 94 8D 96 94 69
9878 : AD A4 94 8D 94 94 AD A1 36
```

Listing 1. Hires-3 (Fortsetzung)

```
9880 : 94 8D 97 94 20 47 96 60 AB
9888 : A9 00 8D A8 94 20 FD AE 49
9890 : 20 EB B7 8E A1 94 A5 14 E3
9898 : 8D 9F 94 A5 15 8D A0 94 38
98A0 : 20 FD AE 20 EB B7 8E A4 6E
98A8 : 94 A6 14 8E A2 94 A5 15 F6
98B0 : 8D A3 94 AD A4 94 CD A1 53
98B8 : 94 B0 09 AE A1 94 8E A4 FF
98C0 : 94 8D A1 94 AE A1 94 8E 7D
98C8 : 9E 94 AD 9F 94 8D 92 94 39
98D0 : AD A0 94 8D 93 94 AD 9E 76
98D8 : 94 8D 94 94 8D 97 94 AD 2E
98E0 : A2 94 8D 95 94 AD A3 94 51
98E8 : 8D 96 94 20 47 96 EE 9E 0C
98F0 : 94 AC A4 94 CC 9E 94 B0 0B
98F8 : D1 60 A9 00 8D A8 94 20 14
9900 : FD AE 20 EB B7 8E A7 94 92
9908 : A5 14 8D A5 94 A5 15 8D B5
9910 : A6 94 20 FD AE 20 8A AD 3A
9918 : A0 94 A2 79 20 D4 BB 20 B2
9920 : FD AE 20 8A AD A0 94 A2 45
9928 : 7E 20 D4 BB A9 79 A0 94 75
9930 : 20 50 B8 20 2B BC 30 0A 18
9938 : A9 79 A0 94 20 A2 BB 4C F7
9940 : 49 99 A9 7E A0 94 20 A2 04
9948 : BB A9 BC A0 B9 20 0F BB 6B
9950 : A2 88 A0 94 20 D4 BB A9 DC
9958 : 20 8D 84 94 A9 00 8D 83 CA
9960 : 94 8D 85 94 8D 86 94 8D 29
9968 : 87 94 20 FD AE 20 8A AD 73
9970 : A0 94 A2 8D 20 D4 BB A9 9F
9978 : 83 A0 94 20 A2 BB 20 6B D4
9980 : E2 A9 79 A0 94 20 28 BA 0A
9988 : A9 11 A0 BF 20 67 B8 20 3A
9990 : 9B BC 18 A5 65 6D A5 94 C6
9998 : 85 14 A5 64 6D A6 94 85 87
99A0 : 15 A9 83 A0 94 20 A2 BB CB
99A8 : 20 64 E2 A9 7E A0 94 20 68
99B0 : 28 BA A9 11 A0 BF 20 67 19
99B8 : B8 20 9B BC 18 A5 65 6D 1E
99C0 : A7 94 AA 20 9B 95 A9 83 74
99C8 : A0 94 20 A2 BB A9 88 A0 7B
99D0 : 94 20 67 B8 A2 83 A0 94 57
99D8 : 20 D4 BB A9 8D A0 94 20 F7
99E0 : 50 B8 20 2B BC 10 90 60 49
99E8 : A9 00 8D A8 94 20 FD AE A9
99F0 : 20 EB B7 8E 94 94 A5 14 72
99F8 : 8D 92 94 A5 15 8D 93 94 DD
9A00 : 20 FD AE 20 8A AD A0 94 90
9A08 : A2 79 20 D4 BB 20 FD AE 1C
9A10 : 20 8A AD A0 94 A2 7E 20 8D
9A18 : D4 BB 20 FD AE 20 8A AD 03
9A20 : A0 94 A2 8D 20 D4 BB 20 3C
9A28 : 6B E2 A9 79 A0 94 20 28 1D
9A30 : BA A9 11 A0 BF 20 67 B8 23
9A38 : 20 9B BC 18 A5 65 6D 92 B8
9A40 : 94 8D 95 94 A5 64 6D 93 ED
9A48 : 94 8D 96 94 A9 8D A0 94 8E
9A50 : 20 A2 BB 20 64 E2 A9 7E B5
9A58 : A0 94 20 28 BA A9 11 A0 CE
9A60 : BF 20 67 B8 20 9B BC 18 22
9A68 : A5 65 6D 94 94 8D 97 94 EB
9A70 : 20 47 96 60 AD 0E DD 09 BA
9A78 : 80 8D 0E DD AD 0F DD 29 1B
9A80 : 7F 8D 0F DD A2 03 A9 00 2E
9A88 : 9D 08 DD CA 10 FA 20 9B 8B
9A90 : B7 A9 00 8D 77 94 F8 E0 8F
9A98 : 00 F0 0F CA 18 69 01 C9 92
9AA0 : 60 90 F4 EE 77 94 A9 00 26
9AA8 : F0 ED 8D 78 94 D8 20 ED 6E
9AB0 : F6 F0 10 AD 0A DD CD 77 8E
9AB8 : 94 90 F3 AD 09 DD CD 78 EE
9AC0 : 94 90 EB 60 20 FD AE 20 90
9AC8 : D4 E1 A2 00 A0 C0 A9 00 EC
9AD0 : 85 FD A9 A0 85 FE A5 01 BB
9AD8 : 48 29 FE 78 85 01 A9 FD 86
9AE0 : 20 D8 FF 68 85 01 58 60 FC
9AE8 : 20 FD AE 20 D4 E1 A9 61 7C
9AF0 : 85 B9 A9 00 4C D5 FF A9 83
9AF8 : FF 8D A8 94 4C 95 95 A9 96
9B00 : FF 8D A8 94 4C 21 96 A9 FE
9B08 : FF 8D A8 94 4C D7 97 A9 C0
9B10 : FF 8D A8 94 4C 8D 98 A9 79
9B18 : FF 8D A8 94 4C FF 98 A9 15
9B20 : FF 8D A8 94 4C ED 99 A9 90
9B28 : 01 A0 3F 20 91 B3 A2 62 53
9B30 : A0 94 20 D4 BB 20 FD AE CF
9B38 : 20 8A AD A2 6C A0 94 20 BB
9B40 : D4 BB 20 FD AE 20 8A AD 2B
9B48 : 20 0C BC A9 6C A0 94 20 31
9B50 : A2 BB 20 53 B8 A9 62 A0 E6
9B58 : 94 20 0F BB A2 62 A0 94 20
9B60 : 20 D4 BB A0 C7 20 A2 B3 5D
9B68 : A2 67 A0 94 20 D4 BB 20 50
9B70 : FD AE 20 8A AD A2 79 A0 35
9B78 : 94 20 D4 BB 20 FD AE 20 B6
9B80 : 8A AD A2 71 A0 94 20 D4 90
9B88 : BB A9 79 A0 94 20 A2 BB D6
9B90 : A9 71 A0 94 20 50 B8 A9 67
9B98 : 67 A0 94 20 0F BB A2 67 A1
9BA0 : A0 94 20 D4 BB 60 A9 00 92
9BA8 : 8D A8 94 20 FD AE 20 8A 9D
9BB0 : AD A9 6C A0 94 20 5B BC 92
9BB8 : C9 FF F0 47 20 0C BC A9 4F
9BC0 : 6C A0 94 20 A2 BB 20 53 D5
9BC8 : B8 A9 62 A0 94 20 28 BA 62
9BD0 : 20 9B BC A5 65 85 14 A5 C0
9BD8 : 64 85 15 20 FD AE 20 8A 33
9BE0 : AD A9 71 A0 94 20 5B BC 03
9BE8 : C9 01 F0 1D A9 71 A0 94 E3
9BF0 : 20 50 B8 A9 67 A0 94 20 AA
9BF8 : 28 BA 20 9B BC A5 65 AA DD
9C00 : 4C A0 95 20 FD AE 20 8A F1
9C08 : AD 60 FF A9 00 8D A8 94 53
9C10 : 20 FD AE 20 8A AD A9 6C 74
9C18 : A0 94 20 5B BC C9 FF D0 32
9C20 : 0B A9 00 8D 92 94 8D 93 DD
9C28 : 94 4C 4D 9C 20 0C BC A9 72
9C30 : 6C A0 94 20 A2 BB 20 53 45
9C38 : B8 A9 62 A0 94 20 28 BA D2
```

```
9C40 : 20 9B BC A5 65 8D 92 94   48
9C48 : A5 64 8D 93 94 20 FD AE   95
9C50 : 20 8A AD A9 71 A0 94 20   05
9C58 : 5B BC C9 01 D0 08 A9 00   98
9C60 : 8D 94 94 4C 7C 9C A9 71   1C
9C68 : A0 94 20 50 B8 A9 67 A0   1C
9C70 : 94 20 28 BA 20 9B BC A5   93
9C78 : 65 8D 94 94 20 FD AE 20   48
9C80 : 8A AD A9 6C A0 94 20 5B   BF
9C88 : BC C9 FF D0 0B A9 00 8D   5C
9C90 : 95 94 8D 96 94 4C B9 9C   71
9C98 : 20 0C BC A9 6C A0 94 20   81
9CA0 : A2 BB 20 53 B8 A9 62 A0   36
9CA8 : 94 20 28 BA 20 9B BC A5   CB
9CB0 : 65 8D 95 94 A5 64 8D 96   B5
9CB8 : 94 20 FD AE 20 8A AD A9   12
9CC0 : 71 A0 94 20 5B BC C9 01   6F
9CC8 : D0 08 A9 00 8D 97 94 4C   87
9CD0 : E8 9C A9 71 A0 94 20 50   6F
9CD8 : B8 A9 67 A0 94 20 28 BA   B3
9CE0 : 20 9B BC A5 65 8D 97 94   FC
9CE8 : 4C 47 96 A9 00 8D A8 94   EB
9CF0 : 20 FD AE 20 8A AD A9 6C   54
9CF8 : A0 94 20 5B BC C9 FF D0   12
9D00 : 0B A9 00 8D 9F 94 8D A0   A8
9D08 : 94 4C 2D 9D 20 0C BC A9   6A
9D10 : 6C A0 94 20 A2 BB 20 53   25
9D18 : B8 A9 62 A0 94 20 28 BA   B2
9D20 : 20 9B BC A5 65 8D 9F 94   5C
9D28 : A5 64 8D A0 94 20 FD AE   16
9D30 : 20 8A AD A9 71 A0 94 20   E5
9D38 : 5B BC C9 01 D0 08 A9 00   78
9D40 : 8D A1 94 4C 5C 9D A9 71   89
9D48 : A0 94 20 50 B8 A9 67 A0   FC
9D50 : 94 20 28 BA 20 9B BC A5   73
9D58 : 65 8D A1 94 20 FD AE 20   6C
9D60 : 8A AD A9 6C A0 94 20 5B   9F
9D68 : BC C9 FF D0 0B A9 00 8D   3C
9D70 : A2 94 8D A3 94 4C 99 9D   81
9D78 : 20 0C BC A9 6C A0 94 20   61
9D80 : A2 BB 20 53 B8 A9 62 A0   16
9D88 : 94 20 28 BA 20 9B BC A5   AB
9D90 : 65 8D A2 94 A5 64 8D A3   F2
9D98 : 94 20 FD AE 20 8A AD A9   F2
9DA0 : 71 A0 94 20 5B BC C9 01   4F
9DA8 : D0 08 A9 00 8D A4 94 4C   D0
9DB0 : C8 9D A9 71 A0 94 20 50   AF
9DB8 : B8 A9 67 A0 94 20 28 BA   93
9DC0 : 20 9B BC A5 65 8D A4 94   10
9DC8 : 4C FD 97 00 00 00 00 00   F9
```

Listing 1. Hires-3 (Schluß)

Listing 2. Dieses Programm testet Hires-3 und demonstriert einige Grafik-Befehle (zum Eintippen bitte den Beitrag »Checksummer 64« beachten)

```
1 REM ***********************************       <250>
2 REM *                                 *       <229>
3 REM *         GRAFIK - DEMO           *       <236>
4 REM *              ZUM                *       <227>
5 REM *  TESTEN DES ERSTEN TEILS VON    *       <220>
6 REM *                                 *       <233>
7 REM *          H I R E S - 3          *       <197>
8 REM *                                 *       <235>
9 REM *      (H.PONNATH HH 1984)        *       <045>
10 REM**********************************       <003>
20 POKE 52,128:POKE 56,128:SYS 37498
   :REM EINSCHALTEN DER NEUEN BASIC-BEFEHLE
                  <138>
25 REM ---------- DER LIN-BEFEHL -------        <054>
30 DEF FN A(X)=(Y2-Y1)*(X-X1)/(X2-X1)+Y1
   :DEF FN B(Z)=(Y4-Y3)*(Z-X3)/(X4-X3)+Y3       <043>
35 DEF FN C(I)=X1+I*(X2-X1)/10
   :DEF FN D(I)=X3+I*(X4-X3)/10                 <130>
40 DEF FN E(I)=X4-I*(X4-X3)/10                  <065>
45 X1=10:X2=100:X3=190:X4=310:Y1=10:Y2=170
   :Y3=195:Y4=15                                <201>
50 HFL,1,6                                      <203>
55 FOR I=0 TO 10 STEP.5:X=FN C(I):Z=FN D(I)
   :LIN,X,FN A(X),Z,FN B(Z):NEXT                <223>
60 LIN,X1,Y1,X2,Y2:LIN,X3,Y3,X4,Y4              <244>
65 PAU,5                                        <136>
70 HOF:PRINT CHR$(147)"DAS WAR DER LIN-BEFEHL"
   :PRINT:PRINT"PAU,HFL U. HOF FUNKTIONIEREN
    AUCH                                        <087>
75 PAU,5                                        <146>
80 REM ----------- DER REC-BEFEHL ------        <100>
85 LOE:HAN                                      <070>
90 DEF FN Z(X)=-3.39098E-3*X↑2+1.053668*X+8.9497
   :X1=200:Y1=198:X2=310:Y2=5                   <117>
95 X3=2:X4=309:E=60                             <121>
100 FOR I=0 TO E:XU=X2-I*(X2-X1)/E:YU=FN A(XU)
   :XO=X4-I*(X4-X3)/E:YO=FN Z(XO)               <138>
105 REC,XO,YO,XU,YU:NEXT I:PAU,3                <033>
110 FAR,6,1:PAU,2:FAR,0,7:PAU,1:FAR,2,8:PAU,1
   :LOE:PAU,1:HOF:PRINT                         <166>
115 PRINT"DAS WAREN DIE BEFEHLE REC,FAR UND
    LOE":PAU,5                                  <137>
120 REM -- PKT,BLO,CIR,LBK UND LRE -----        <019>
125 HAN:BLO,40,100,160,150:BLO,180,100,200,150
   :BLO,200,120,250,150                         <111>
130 BLO,240,100,250,120:LBK,44,102,68,120
   :LBK,72,102,98,120:LBK,102,102,128,120       <249>
135 LBK,132,102,156,120:LBK,182,102,198,148
   :LRE,44,122,68,148                           <109>
140 LRE,132,122,156,148:LIN,160,145,180,145
   :CIR,45,155,5,5,2*π                          <243>
145 CIR,55,155,5,5,2*π:CIR,155,155,5,5,2*π
   :CIR,145,155,5,5,2*π                         <120>
150 CIR,185,155,5,5,2*π:CIR,195,155,5,5,2*π
   :CIR,245,155,5,5,2*π                         <182>
155 LIN,0,160,319,160:LIN,0,163,319,163         <247>
160 FOR K=1 TO 10:FOR J=1 TO 235                <122>
165 V=INT(RND(1)*20)+235-J:W=INT(RND(1)*100-J/3)
   :IF W>0 AND V>0 THEN:PKT,V,W:NEXT            <067>
170 NEXT K:PAU,5:HOF                            <015>
175 PRINT:PRINT"DIESER ZUG FUHR MIT
   :PKT,BLO,CIR,LBK,LRE":PAU,5                  <001>
180 REM ------ RAD UND LRA -------------        <055>
185 HFL,0,6:FOR L=0 TO 3*πSTEPπ/30
   :RAD,160,100,INT(10*L),INT(10*L),L           <147>
190 NEXT L:PAU,4:FOR L=0 TO 3*πSTEPπ/15
   :LRA,160,100,INT(7*L),INT(7*L),L:NEXT L <013>
195 PAU,4:HOF:PRINT:PRINT"SOWOHL RAD ALS AUCH
    LRA FUNKTIONIEREN":PAU,4                    <254>
200 REM ------ LPK,LLN,LKR -------------        <188>
205 POKE 53280,0                                <194>
210 HFL,6,14:BLO,20,10,300,190:X(1)=60:X(2)=120
   :X(3)=170:X(4)=200:X(5)=201                  <244>
215 X(6)=240:X(7)=260:Y(1)=65:Y(2)=42:Y(3)=70
   :Y(4)=100:Y(5)=135:Y(6)=150                  <019>
220 Y(7)=117:FOR I=1 TO 7:LKR,X(I),Y(I),2,2,2*π
   :LKR,X(I),Y(I),1,1,2*π:NEXT I                <023>
225 FOR I=2 TO 7:LLN,X(I-1),Y(I-1),X(I),Y(I)
   :NEXT I:LLN,X(7),Y(7),X(4),Y(4)              <013>
230 FOR I=0 TO 500:V=INT(RND(1)*280)+20
   :W=INT(RND(1)*180)+10:LPK,V,W:NEXT I:PAU,3
                  <034>
235 POKE 53280,14:HOF:PRINT:PRINT"DER GROSSE
    WAGEN WURDE GEBILDET MIT"                   <231>
240 PRINT"DEN BEFEHLEN:LLN,LPK UND LKR":PRINT
   :PRINT:POKE 646,1:PRINT"ALLES O.K!"          <197>
245 POKE 646,14:END                             <119>
```

Comal — Eine

In dieser Folge unserer Comal-Einführung befassen wir uns mit einem der interessantesten Aspekte dieser Programmiersprache, nämlich der Grafik.

Die Comal-Grafik kann ihre Herkunft von einer anderen Programmiersprache nicht verheimlichen. Befehle wie »SHOWTURTLE« und »TURTLESIZE« sind verräterisch: Die Sprache Logo mit ihrer bekannten Turtle-Grafik stand neben Basic und Pascal Pate beim Entwurf von Comal.

Die Bezeichnung Turtle-Grafik stammt noch aus den Anfängen von Logo. Turtle ist das englische Wort für »Schildkröte« und bezeichnete ursprünglich eine mechanische Schildkröte, die sich per Computersteuerung über ein großes Blatt Papier bewegte und dabei mit einem Schreibstift eine sichtbare Spur hinterließ. Zur Steuerung dieser »Schildkröte« wurde eine spezielle Programmiersprache entwickelt, mit Befehlen wie »FORWARD«, »BACK«, »LEFT« etc. Das funktionierte einige Zeit ganz gut, aber dann tauchten die ersten Mikrocomputer mit Grafikbildschirmen auf, und die Schildkröte hauchte ihr mechanisches Leben aus und degenerierte zu einem kleinen Dreieck auf einem Grafikbildschirm. Doch die grafische Programmiersprache Logo war geboren und ist seitdem immer weiterentwickelt worden.

Das Konzept der Turtle-Grafik wurde vollständig in Comal integriert (Tabelle 1). Daneben gibt es auch spezielle Kommandos zum Zeichnen einzelner Punkte oder Linien, und zwar unabhängig von der jeweiligen Turtleposition. Überhaupt wird der gesamte Bereich von Grafik, Farbe und Bildschirmansteuerung beim C 64 durch Comal-Befehle abgedeckt (Tabelle 2), man muß sich nicht wie im V.2-Basic mit merkwürdigen PEEKs und POKEs herumschlagen.

Comal kennt drei verschiedene Bildschirmmodi, nämlich den normalen Textmodus (»SETTEXT«), den hochauflösenden Grafikmodus (»SETGRAPHIC 0«) und den Mehrfarben-Grafikmodus (»SETGRAPHIC 1«). Der Textmodus wird auch mit der Funktionstaste f1 eingeschaltet, während f5 die Grafik einschaltet. Im hochauflösenden Modus kann mit f3 zusätzlich noch ein Textfenster eingeblendet werden, was beim Experimentieren mit der Turtle-Grafik recht nützlich ist. Die oberen beiden Bildschirmzeilen bilden dann das Textfenster, mit dessen Hilfe man beispielsweise im Direktmodus mit entsprechenden Befehlen die Turtle steuern kann.

Geben Sie einmal im Direktmodus den Befehl »CLEAR« ein (um den Grafikbildschirm zu löschen) und schalten Sie danach durch Drücken der f3-Taste in die hochauflösende Grafik mit Textfenster um. Sie sehen in der Mitte des ansonsten leeren Bildschirms ein kleines weißes Dreieck, die Turtle. In der linken oberen Bildschirmecke blinkt der Cursor und zeigt damit an, daß die Turtle auf Befehle von Ihnen wartet.

Anweisung	Bedeutung
BACKGROUND n	Hintergrundfarbe n wählen
BORDER n	Rahmenfarbe n wählen
CLEAR	Grafikbildschirm löschen
DRAWTO x,y	Zeichnet eine Linie zum Punkt x,y
FILL	Flächen mit Farbe ausfüllen
FRAME x1,x2,y1,y2	Window festlegen
FULLSCREEN	Ganzer Bildschirm für Grafik
MOVETO x,y	Grafik-Cursor auf x,y setzen
PLOT x,y	Grafikpunkt setzen
PLOTTEXT x,y,text$	Text in Grafikbildschirm einblenden
SETGRAPHIC	Grafik-Modus wählen
SETTEXT	Auf Textbildschirm umschalten
SPLITSCREEN	In Grafikbildschirm zwei Textzeilen reservieren

Tabelle 2. Allgemeine Bildschirm- und Grafikbefehle

Anweisung	Bedeutung
FORWARD n	Turtle n Schritte vorwärts
BACK n	Turtle n Schritte rückwärts
HOME	Turtle auf Grundposition zurücksetzen
LEFT w	Turtle um w Grad links drehen
RIGHT w	Turtle um w Grad rechts drehen
SETXY x,y	Turtle auf die Koordinaten x,y setzen
SETHEADING w	Turtle-Richtung absolut festlegen (w=0 bedeutet, daß Turtle nach oben zeigt)
SHOWTURTLE	Turtle sichtbar machen
HIDETURTLE	Turtle unsichtbar machen
PENUP	Turtleweg wird nicht mitgezeichnet
PENDOWN	Turtleweg wird mitgezeichnet
PENCOLOR n	Zeichenfarbe n wählen
TURTLESIZE n	Turtlegröße auf n Punkte festlegen

Tabelle 1. Befehle für Turtle-Grafik

So wird die »Schildkröte« bewegt

Geben Sie jetzt »FORWARD 50« ein. Die Turtle bewegt sich damit um 50 Einheiten vorwärts und zieht dabei eine Linie längs ihres Weges. Mit »LEFT 90« erreichen Sie eine Drehung der Turtle um 90 Grad nach links. Wenn Sie jetzt nochmals »FORWARD 50« eingeben, bewegt sich die Turtle in die neue Rich-

Einführung (3)

Anweisung	Bedeutung
DATACOLLISION n,b	Sprite-Hintergrund-Kollisionsabfrage für Sprite n (die Variable b wird 1, falls Kollision, sonst b=0)
SPRITECOLLISION n,b	Sprite-Sprite-Kollisionsabfrage für Sprite n (b wird 1, falls Kollision, sonst b=0)
DEFINE d,x$	Der Inhalt der Stringvariablen x$ definiert ein Sprite mit Definitionsnummer d
IDENTIFY n,d	Ordnet dem mit Nummer d definierten Sprite die Spritenummer n zu
PRIORITY n,p	Hintergrund hat Priorität vor Sprite n (p=0), umgekehrt für p=1
SPRITEBACK a,b	Setzt a und b als zusätzliche Farben für Multicolor-Sprites
HIDESPRITE n	Sprite n nicht anzeigen
SHOWSPRITE n	Sprite n anzeigen
SPRITECOLOR n,c	Farbe c für Sprite n wählen
SPRITEPOSITION n,x,y	Sprite n auf Position x,y setzen
SPRITESIZE n,x,y	Sprite n in x- oder y-Richtung vergrößern

Tabelle 3. Sprite-Befehle

tung. Soll eine Bildschirmstelle angefahren werden, ohne eine Linie dorthin zu ziehen, dann wird mit dem Kommando »PENUP« einfach der symbolische Schreibstift von der Zeichenfläche abgehoben. Die Turtle kann dann nach Belieben über den gesamten Bildschirm dirigiert werden, ohne Spuren zu hinterlassen. Aber keine Angst, nach »PENDOWN« zeichnet sie wieder.

Wenn Sie es gerne etwas bunter hätten, bitte sehr. »PENCOLOR« wählt die Schreibfarbe der Turtle, »BORDER« und »BACKGROUND« wählen Rahmen- und Hintergrundfarbe. Erscheint Ihnen die Turtle zu gut genährt, dann können Sie zum Beispiel mit »TURTLESIZE 5« bedenkenlos etwas Speck entfernen. Und sollte Ihnen die Turtle insgesamt nicht ganz geheuer sein, dann tippen Sie einfach »HIDETURTLE« in Ihren Computer ein. Die »Schildkröte« wird sich daraufhin schmollend von der Bildfläche zurückziehen, aber zum Glück wird ihr Wirken dadurch nicht behindert: Aus dem Unsichtbaren befolgt sie weiter Ihre Befehle und zieht fleißig ihre Linien.

Kleine Programme mit großer Wirkung

Natürlich ist die Grafikerzeugung mit dieser sehr direkten Methode auf die Dauer entschieden zu langwierig. Wenn Sie das kleine Programm aus Listing 1 einmal ausprobieren, werden Sie feststellen, daß effektvolle Grafikprogramme in Comal weder besonders komplex noch langsam zu sein brauchen. Experimentieren Sie ruhig mit ähnlichen Programmen; Sie werden sehr schnell ein gewisses Gespür dafür entwickeln, wie Sie durch Wiederholung einfacher Grundfiguren mit jeweils einem etwas geänderten Parameter interessante grafische Effekte erzielen können.

Zugabe: Sprites in Comal programmiert

Der große Vorteil der Turtle-Grafik gegenüber dem Zeichnen von Linien nach festen Koordinaten ist ja gerade die erstaunliche Einfachheit, mit der sich recht komplexe Strukturen vom Standpunkt der »Schildkröte« aus darstellen. Mit Comal können Sie wirklich eine »Reise durch die Wunderwelt der Grafik« antreten — auch ohne sich mit Bits und Bytes abzuplagen und mit komplizierten mathematische Formeln zu jonglieren.

Die Turtle-Grafik ist nur die eine Seite der grafischen Fähigkeiten des C 64 - Comal. Es stehen nämlich zusätzlich eine Reihe von leistungsfähigen Anweisungen zur Erzeugung und Kontrolle von Sprites zur Verfügung.

Auch bei Sprites wird zwischen Hochauflösung (Hires) und Mehrfarbendarstellung (Multicolor) unterschieden. Ein Hires-Sprite besteht bekanntlich aus 24 x 21 Punkten oder aus 21 Reihen zu je drei Byte. Bei Multicolor-Sprites haben wir nur 12 x 21 Punkte, dafür benötigt jeder Bildpunkt aber intern zwei Bit, da vier verschiedene Farben pro Punkt möglich sind. In beiden Fällen kann die in einem Sprite enthaltene Information in 63 Bytes dargestellt werden. Comal verwendet ein zusätzliches Byte, das angibt, ob es sich um ein Hires-Sprite (Byte 64 = 0) oder um ein Multicolor-Sprite (Byte 64 <> 0) handelt. Beide Spritearten werden im übrigen

```
0001 //
0002 // Grafik-Demo
0003 //
0010 print chr$(147)
0020 pencolor 1
0030 input " Abstand:   1▮▮": abstand
0040 input " Winkel :  89▮▮▮": winkel
0050 input " Inkrement: 1▮▮": i
0060 setgraphic 0
0070 while abstand<365 do
0080  forward abstand
0090  left winkel
0100  abstand:=abstand+i
0110 endwhile
```

Listing 1. Eine kleine Demonstration der Turtle Grafik

Comal — Eine Einführung (3)

```
0010 // ----------------------
0020 // Turtle- und Sprite-Demo
0030 // ----------------------
0040 //
0050 //
0060 settext
0070 input "Rahmenfarbe: ": rahmen
0080 border rahmen
0090 input "Hintergrundfarbe: ": grund
0100 background grund
0110 setgraphic 0
0120 dim sprite$ of 63
0130 dim ein$ of 4
0140 //
0150 // Turtle initialisieren
0160 //
0170 penup
0180 setxy 160,160
0190 pendown
0200 turtlesize 10
0210 showturtle
0220 //
0230 // Einfaches Sprite-Bild erzeugen
0240 //
0250 sprite$:=""
0260 ein$:=chr$(255)+chr$(0)+chr$(255)
0270 y:=50
0280 for i:=1 to 63/3 do
0290  sprite$:=sprite$+ein$
0300 endfor i
0310 //
0320 // Zwei Sprites definieren
0330 //
0340 define 0,sprite$+chr$(0)
0350 sprite$(62):=chr$(255)
0360 sprite$(2):=chr$(255)
0370 define 1,sprite$+chr$(0)
0380 //
0390 // Vier Sprites aus zwei Bildern
0400 //
0410 identify 0,0
0420 identify 1,0
0430 identify 2,1
0440 identify 3,1
0450 //
0460 // Spritefarbe setzen
0470 //
0480 spritecolor 0,0
0490 spritecolor 1,2
0500 spritecolor 2,7
0510 spritecolor 3,1
0520 //
0530 // Demo laeuft bis Taste gedrueckt wird
0540 //
0550 repeat
0560  pencolor rnd(0,15)
0570  bewegung
0580  penup
0590  right 45
0600  forward 80
0610  pendown
0620 until key$<>chr$(0) // Taste gedrueckt ?
0630 //
0640 // Sprites und Turtle bewegen
0650 //
0660 proc bewegung
0670  //
0680  // Spritegroesse zufaellig aendern
0690  //
0700  spritesize 0,rnd(0,1),rnd(0,1)
0710  spritesize 1,rnd(0,1),rnd(0,1)
0720  spritesize 2,rnd(0,1),rnd(0,1)
0730  spritesize 3,rnd(0,1),rnd(0,1)
0740  for x:=1 to 300 step 5 do
0750   //
0760   // Turtle bewegen
0770   //
0780   left 88
0790   forward 50
0800   //
0810   // Sprites bewegen
0820   //
0830   spritepos 0,x,y
0840   spritepos 1,300-x,y+50+x/3
0850   spritepos 2,x,y+50+x/3
0860   spritepos 3,300-x,y
0870  endfor x
0880 endproc bewegung
```

Listing 2. Turtle-Grafik und Sprites: So einfach und übersichtlich ist die Programmierung

mit völlig gleichartigen Befehlen angesprochen.

Zur Spriteerzeugung dient der Befehl »DEFINE«. Als Parameter wird eine Definitions-Nummer sowie ein String angegeben. Die Definitions-Nummer kann irgendeine Zahl von 0 bis 95 sein, denn Comal kann bis zu 96 verschiedene Spritemuster verwalten. Der anzugebende String muß eine Länge von 64 Zeichen haben und das Punktmuster des Sprites enthalten. Das Beispielprogramm in Listing 2 zeigt, wie man in Comal mit Sprites umgeht.

Wichtig ist es, genau zwischen der Definitions-Nummer eines Sprites und der eigentlichen Spritenummer zu unterscheiden. Es können hardwarebedingt ja immer nur acht Sprites gleichzeitig auf dem Bildschirm sein. Wird die Turtle benutzt, die selber ein Sprite ist, dann verringert sich diese Anzahl auf sieben.

Mit dem »IDENTIFY«-Kommando wird die Beziehung zwischen Sprite-Nummer und Definitions-Nummer hergestellt. »IDENTIFY 1,30« bewirkt beispielsweise, daß das Sprite mit der Definitions-Nummer 30 als Sprite 1 auf dem Bildschirm dargestellt wird. So kann sehr schnell zwischen verschiedenen Sprites umgeschaltet werden, was zur Erzeugung von zeichentrickähnlichen Effekten benutzt werden kann.

Wie aus Tabelle 3 zu ersehen ist, gibt es eine Vielzahl von weiteren Comal-Befehlen speziell zur Sprite-Kontrolle. Die Wirkungsweise solcher Anweisungen wie »SPRITECOLOR« oder »SPRITEPOS« dürfte wohl jedem verständlich sein, der sich schon einmal von Basic aus an die Spriteprogrammierung gewagt hat.

Comal oder Basic?

Wie die drei Folgen unserer kleinen Comal-Einführung hoffentlich gezeigt haben, hat Comal dem Commodore V.2-Basic einiges voraus. Die von Pascal entlehnte Programmstrukturierung mittels Prozeduren und die aus Logo stammende Turtle-Grafik haben sich mit dem aus Basic übernommenen Konzept einer benutzerfreundlichen, interaktiven Programmiersprache zu einer gelungenen Synthese zusammengefügt. Die hier besprochene Comal-Version 0.14 ist dabei nur als eine Art Vorabversion zu werten. Eine nochmals wesentlich erweiterte und verbesserte Version 2.0 wird von Commodore Dänemark bereits als Steckmodul (64 KByte ROM, 30 KByte RAM frei) vertrieben. Es ist zu hoffen, daß auch Commodore Deutschland diese zukunftsweisende Programmiersprache bald anbieten wird. (ev)

Info: Comal-Literatur:
Borge Christensen, Heiko Wolgast: Comal 0.14-Handbuch, 80 Seiten, Verlag Schmidt & Klaunig, Ringstraße, 2300 Kiel.
Borge Christensen: Strukturierte Programmierung mit Comal, 230 Seiten, Verlag R. Oldenbourg, München, ISBN 3-486-26901-1

Assembler ist keine Alchimie (Teil 6)

In der vorangegangenen Ausgabe haben wir die relative und die Zeropage-Adressierung kennengelernt. Heute kommt die indizierte Adressierung dran und natürlich sehen wir uns wieder einige neue Assembler-Befehle an. Wir werden uns einige Gedanken machen über die sogenannten Fließkommazahlen und den Basic-Befehl USR. Auch die Speicherorganisation unseres Computers soll uns nochmal beschäftigen.

Zunächst die indizierte Adressierung. Indizieren heißt, etwas mit einem Index, also einem Zeichen oder einer Nummer, zu versehen. Beispielsweise bezeichnet man in der Mathematik die beiden Lösungen einer quadratischen Gleichung häufig als X1 und X2. Dabei ist dann die Ziffer (1 oder 2) der Index und X ist eine indizierte Größe. Man geht also aus von einer festgelegten Grundmenge (Lösungsmenge X) und trifft durch den Index eine weitere Unterscheidung.

So ähnlich können wir uns auch die Funktion der indizierten Adressierung bei der Assembler-Programmierung vorstellen. Nehmen wir als Beispiel den Befehl
LDA 1500,X
Man spricht hier von einer absolut-X-indizierten Adressierung. Das Assemblerwort LDA ist uns bekannt: Lade den Akku. Woher soll der für den Akku bestimmte Inhalt geholt werden? Aus der Speicherzelle, die sich durch 1500 plus Inhalt des X-Registers ergibt. Steht also im X-Register zum Zeitpunkt des Befehlsaufrufes eine 5, dann wird der Akku aus Speicherzelle 1500+5, also 1505, geladen. Das X-Register kann Werte von 0 bis $FF (dez. 255) enthalten. Die Ähnlichkeit sieht also so aus:

Aus einer Gesamtmenge von 256 Adressen, die durch die Anfangsadresse (bei unserem Beispiel 1500) und die möglichen 256 Belegungen des X-Registers festgelegt sind (die Grundmenge), werden je nach X-Registerinhalt einzelne Adressen unterschieden und adressiert. Das X-Register fungiert dabei als ein Index, weswegen man auch oft die Bezeichnung »Index-Register X« in der Literatur findet.

Ebenfalls als Index-Register kann das Y-Register dienen, was zum Beispiel zum Befehl
LDX 1500,Y
führen kann. Dies ist dann eine absolut-Y-indizierte Adressierung.

Befehl	Indizierte Adressierung			
	absolut		Null-Seite-absolut	
	X	Y	X	Y
Folge 2				
LDA	+	+	+	—
LDX	—	+	—	+
LDY	+	—	+	—
STA	+	+	+	—
STX	—	—	—	+
STY	—	—	+	—
RTS	/	/	/	/
Folge 3				
INX	/	/	/	/
INY	/	/	/	/
INC	+	—	+	—
DEX	/	/	/	/
DEY	/	/	/	/
DEC	+	—	+	—
SED	/	/	/	/
CLD	/	/	/	/
BNE	/	/	/	/
Folge 4				
ADC	+	+	+	—
CLC	/	/	/	/
SBC	+	+	+	—
SEC	/	/	/	/
BEQ	/	/	/	/
BCC	/	/	/	/
BCS	/	/	/	/
BMI	/	/	/	/
BPL	/	/	/	/
BVC	/	/	/	/
BVS	/	/	/	/
Folge 5				
CMP	+	+	+	—
CPX	—	—	—	—
CPY	—	—	—	—
Folge 6				
BIT	—	—	—	—
CLV	/	/	/	/
NOP	/	/	/	/
TAX	/	/	/	/
TAY	/	/	/	/
TXA	/	/	/	/
TYA	/	/	/	/
JMP	—	—	—	—
JSR	—	—	—	—

+ anwendbar
— nicht erlaubt
/ weder absolute noch Zeropage-Adressierung möglich

Tabelle 1. Anwendbarkeit der indizierten Adressierungsarten auf die bisher gelernten Assembler-Befehle.

Genauso wie man die normale absolute Adresse (also zum Beispiel 1500) als Basis der Indizierung durch das X- oder das Y-Register verwenden kann, ist das auch mit eine Zeropage-Adresse möglich. So gibt es zum Beispiel die Befehle
LDY 2B,X
oder
STX 19,Y
Man nennt diese Art der Adressierung dann Zeropage-absolut-X-indiziert beziehungsweise -Y-indiziert.

Weil die Zeropage aber nur 256 Adressen umfaßt, andererseits jedoch die Indexregister auch 256 Werte annehmen können, kann es geschehen (wenn man nicht aufpaßt), daß die Summe aus der Basisadresse (zum Beispiel $2B) und dem Indexregisterinhalt größer als 256 wird. Wenn zum Beispiel in dem Befehl
LDA FE,X
der X-Registerinhalt 2 beträgt, ergäbe sich $FE+$02=$0100. In diesem Fall wird aber nicht der Inhalt von $0100 in den Akku geladen, sondern der Befehl spricht die Speicherstelle $00 an. Der Grund dafür liegt in der Tatsache, daß unser Prozessor den Befehl als 2-Byte-Befehl interpretiert — das 2. Byte ist die Zeropageadresse, die sich als Summe ergibt — und deswegen nur das LSB der Adresse beachtet. Von $0100 ist das LSB aber $00. Mit anderen Worten: Die Zero-page-absolut-indizierten Befehle lassen einen Zugriff nur auf die Zeropage selbst zu. Dieses Verhalten muß man beim Programmieren beachten.

Wir wollen nochmal zusammenfassen. Vier neue Adressierungsarten haben wir kennengelernt:

Absolut-X-indiziert
zum Beispiel LDA 1500,X
Absolut-Y-indiziert
zum Beispiel LDA 1500,Y
Zero-page-absolut-X-indiziert
zum Beispiel LDA 2B,X
Zero-page-absolut-Y-indiziert
zum Beispiel LDX 2B,Y

Die Verwendung des Y-Registers als Indexregister ist stark eingeschränkt. Nur bei wenigen Befehlen ist sie erlaubt (tatsächlich nur LDX und STX bei Zeropage-absolut-indizierter Adressierung). In der Tabelle 1 sehen Sie, welche bisher behandelten Befehle wie mit der indizierten Adressierung verwendet werden dürfen.

Es gibt noch zwei weitere Arten einer indizierten Adressierung, auf die wir noch zu sprechen kommen werden.

Einige Nachzügler: Die Befehle BIT, CLV, NOP und TAX, TAY, TXA, TYA

Wir wollen noch ein bißchen aufräumen: Ein paar Befehle, die bisher zu keinem Gebiet so richtig paßten, sollen jetzt behandelt werden.

BIT: Dieser Befehl heißt »Bit-Test« und paßt von daher eigentlich zu den in der letzten Ausgabe behandelten Vergleichsbefehlen. Die Behandlung der Flaggen ist aber völlig anders. Nehmen wir das Beispiel
BIT 1500

Folgendes passiert: Der Inhalt der Speicherstelle $1500 wird mit dem Inhalt des Akku UND-verknüpft, das Ergebnis in der Z-Flagge angezeigt und Bit 7 sowie Bit 6 von $1500 in die N- beziehungsweise die V-Flagge übertragen. Weder Akku noch der Inhalt von $1500 verändern sich dabei.

Das ging ein bißchen holterdipolter. Sehen wir uns das jetzt mal ganz langsam Schritt für Schritt an! Zunächst die UND-Verknüpfung. Bit für Bit wird der Akku-Inhalt mit dem Inhalt der adressierten Speicherstelle UND-verknüpft. Dabei gelten folgende Regeln (die Leser der Grafik-Serie kennen das ja schon):

0 UND 0 = 0
0 UND 1 = 0
1 UND 0 = 0
1 UND 1 = 1

Nur dann also, wenn die entsprechenden Bits im Akku und in 1500 gleich 1 sind, ergibt sich bei der UND-Verknüpfung eine 1. Man stellt sowas meist in einer sogenannten Wahrheitstabelle zusammen (Tabelle 2).

UND	0	1
0	0	0
1	0	1

Tabelle 2. Wahrheitstabelle der logischen Verknüpfung UND

Nehmen wir als Beispiel mal an, im Akku stünde $0A und in der Speicherstelle $1500 wäre $09 enthalten. Die UND-Verknüpfung sieht dann so aus:

```
Akku $0A 0000 1010
1500 $09 0000 1001
UND      ---------
         0000 1000
```

Das Ergebnis ist also $08. In der Z-Flagge wird in dem Fall, daß das Ergebnis der UND-Verknüpfung ungleich Null ist (wie hier) eine Null angezeigt, sonst eine 1.

Wir haben in unserem Zahlenbeispiel mit dem BIT-Befehl überprüft, ob die Bits 1 und 3 in Speicherstelle $1500 gelöscht sind. Dazu haben wir in den Akku eine sogenannte Maske (hier also $0A) geladen. Das Ergebnis sagt uns, daß nicht beide Bits gelöscht waren. Wäre der Inhalt von $1500 beispielsweise $10 gewesen (0001 0010), hätten wir in der Z-Flagge eine 1 gefunden. Daher der Name »Bit-Test«: Durch geeignete Maskenwahl kann praktisch jedes Bit überprüft werden. Dabei werden weder der Akku-Inhalt noch der Inhalt der angesprochenen Speicherstelle verändert.

Der BIT-Befehl hat aber noch mehr Auswirkungen: Die Bits 6 und 7 der geprüften Speicherzelle findet man nach Befehlsausführung in zwei Flaggen nochmal:
Bit 7 in der N-Flagge
Bit 6 in der V-Flagge

Damit kann man beispielsweise überprüfen, ob sich am adressierten Ort eine negative Zahl befindet. Alle drei Flaggen können ja nun mit den Branch-Befehlen abgefragt werden. Sie erkennen sicherlich schon, wie vielseitig dieser merkwürdige BIT-Befehl einsetzbar ist.

Adressierbar ist BIT entweder absolut (wie im obigen Beispiel) oder Zeropage-absolut. Je nachdem liegt er dann als 3-Byte- oder als 2-Byte-Befehl vor.

CLV: Dieser Befehl heißt »clear overflow-flag«, also »lösche die Überlauf-Flagge«. Die V-Flagge war — wie Sie sich erinnern werden — unsere rote Ampel bei Rechenoperationen (siehe Ausgabe 12/84, S. 135). Es ist ein 1-Byte-Befehl mit impliziter Adressierung und interessant daran ist, daß es keinen Befehl gibt, der das Gegenteil — also das Setzen der V-Flagge — bewirkt.

NOP: NOP steht für »no operation«, was bedeutet »keine Tätigkeit«. Das ist der Nichtstu-Befehl. Er tut aber doch etwas: Er sorgt dafür, daß der Befehlszähler weitergezählt wird und bewirkt eine Verzögerung von 2 Taktzyklen. NOP ist ein 1-Byte-Befehl mit impliziter Adressierung. Er wird in fertigen Programmen nur selten verwendet: Zur Erzeugung einer kurzen definierten Verzögerung. Meist gebraucht man ihn bei der Erstellung eines Programmes als Platzhalter oder bei der Fehlersuche, um zum Beispiel unerwünschte Sprünge zu ersetzen.

Die Transporteure: TAX, TAY, TXA und TYA

Ab und zu ist es nötig, Registerinhalte untereinander auszutauschen. Viele Dinge (Addition, Subtraktion und so weiter) können nur im Akku geschehen. Wenn wir eine solche Operation beispielsweise mit dem Inhalt des X-Registers ausführen wollen, verschieben wir diesen Inhalt mit dem Befehl TXA. »Transfer X into accumulator« also »übertrage X-Register in den Akku« bedeutet das. Analog verwendet man TYA, um Y-Register-Inhalte in den Akku zu schieben oder für den umgekehrten Weg TAY beziehungsweise TAX (Akkuinhalt ins Y- beziehungsweise ins X-Register schieben). Genau genommen wird nicht übertragen, sondern nur kopiert: Die Register, aus denen verschoben wird, bleiben unverändert. Weil die jeweiligen Zielorte der Verschiebung (Akku, X- oder Y-Register) vom neuen Inhalt überschrieben werden, können sich auch Flaggen ändern. Betroffen sind von dieser Möglichkeit die N- und die Z-Flagge. Alle vier Befehle bestehen aus einem Byte und können natürlich nur implizit adressiert werden.

So springen die Assembler-Alchimisten: JMP, JSR

JMP und JSR entsprechen ungefähr den vom Basic her bekannten Befehlen GOTO und GOSUB.

JMP kommt von »jump to address«, also »springe zur angegebenen Adresse«. Nehmen wir uns wieder ein Beispiel vor:
JMP 1500
bewirkt einen Sprung zur Adresse 1500. Das funktioniert so: In den Programmzähler werden LSB und MSB der Zieladresse geladen. Das war dann auch schon der Sprung, denn der Programmzähler ist der Pfadfinder des Computers: Die Adresse, die dort steht, wird als nächste bearbeitet. Schalten Sie doch mal den SMON ein (oder einen anderen Monitor) und sehen Sie sich das mit folgenden Befehlen an:

```
1400 JMP 1500
```

Dort unterbrechen wird den Computer mit

```
1500 BRK
```

So weit, so gut: Wir starten mit dem SMON-Kommando G 1400 und erhalten eine Registeranzeige mit dem Programmzählerstand 1501. Genau das hatten wir ja erwartet.

Weniger durchschaubar ist das folgende Beispiel:

```
1400  LDA  #00
1402  LDX  #16
1404  STA  1300
1407  STX  1301
140A  JMP  (1300)
```

Dazu gehört dann noch die Programmzeile:

```
1600  BRK
```

Wenn Sie das genauso eingegeben haben und dann mittels G 1400 starten, erhalten Sie eine Registeranzeige mit dem Programmzählerstand 1601.

Schon an der neuen Schreibweise des Argumentes in Zeile 140A werden Sie bemerkt haben, daß hier nicht mehr die normale absolute Adressierung wie zuvor angewendet wird. Dies ist eine neue Form: Die **indirekte Adressierung**. Indirekt deswegen, weil wir nicht mehr direkt die Zieladresse angeben, sondern einen sogenannten Vektor. Ein Vektor besteht aus zwei aufeinander folgenden Speicherzellen (hier also 1300 und 1301), die in der Form LSB/MSB die eigentliche Zieladresse enthalten. Das LSB von $1600 ist $00. Das haben wir über den Akku nach $1300 geladen. Das MSB $16 kam durch das X-Register an seinen Platz $1301:

```
Zieladresse  16    00
             MSB   LSB
             ↑     ↑
Vektor       1301  1300
```

Das ist die Methode der toten Briefkästen, die in Kreisen der Assembler-Alchimisten anscheinend genauso beliebt ist wie bei Agenten. So wie diese im hohlen Baum die Treffpunktanschrift hinterlegt finden, verläßt sich unser Computer auf die Speicherstellen 1300 und 1301 für die Angabe der Zieladresse.

Diese Art der Adressierung ist im wahrsten Sinn des Wortes ein Unikum: Es gibt sie nämlich nur für den JMP-Befehl! Davon wird allerdings dann auch recht häufig Gebrauch gemacht, zum Beispiel im Betriebssystem unseres Computers. Aber darüber und über die Vektoren, die dazu verwendet werden, soll ein andermal berichtet werden.

Wir dürfen nämlich nicht den anderen **Sprungbefehl JSR** vergessen. JSR steht für »jump to subroutine«, was eingedeutscht etwa bedeutet »springe zum Unterprogramm«. Genauso wie in Basic Unterprogramme durch GOSUB (Zeilennummer) aufgerufen werden, kann das auch hier geschehen durch JSR Adresse. Hier ist nur die absolute Adressierung möglich. Das erste Beispiel soll uns zeigen, wie dieser Befehl funktioniert:

```
1400  JSR 1500
```

Dort soll dann erstmal stehen:

```
1500  BRK
```

Noch nicht starten!! Zunächst einmal verzeihen Sie mir diese Programmierer-Todsünde: Aus einem Unterprogramm heraus den Programmablauf zu beenden! Ich werd's auch nie wieder tun. Hier geschieht das nur zu

Lehrzwecken. Was läuft ab: Der Programmzählerinhalt plus 2 wird auf den Stapel gelegt und dann die Adressse 1500 in den Programmzähler geladen. Ebenso kurz wie unklar! Was ist denn ein Stapel? Also langsam, Schritt für Schritt.

Der Sinn von Unterprogrammen ist ja, daß der Computer nach Ende der Bearbeitung wieder ins aufrufende Hauptprogramm zurückkehrt. Er muß sich aber dazu irgendwo merken, von wo aus er zum Unterprogramm gesprungen ist. Dazu verwendet er den Stapel. Das ist ein Speicherbereich ($0100 bis $01FF), der direkt vom Prozessor aus verwaltet wird. Die genaue Architektur und Handhabung dieses »Prozessor-Stack« werden wir noch in einer späteren Folge kennenlernen. Uns soll hier nur interessieren, daß es einen Zeiger gibt, der auf den nächsten freien Platz im Stapel weist und daß dieser Speicher von oben nach unten gefüllt wird (wie in Basic bei den Strings). Wenn Sie mit Hilfe des SMON mal in den Stapel hineinsehen wollen, dann geben Sie doch mal ein M 0100 01FF. Was nun genau bei Ihnen drin steht, ist sehr von der vorherigen Nutzung Ihres Computers abhängig. Der Mikroprozessor nutzt den Stapel bei sehr vielen Tätigkeiten. Es kommt auch nur auf den Teil des Stapels an, der durch den Stapelzeiger als gefüllt bezeichnet wird. Der Stapelzeiger wird beim SMON in der Registeranzeige als SP angezeigt. Wenn Ihr Stapelzeiger (prüfen Sie das doch mal durch Eingabe von R) nun zum Beispiel F6 zeigt, dann bedeutet das, daß alle Stapelplätze von $01F6 an abwärts frei und die oberhalb bis $01FF besetzt sind. Beim Nachsehen mit M 01F0 01FF finden Sie dann beispielsweise:

```
:01F0   20  00  20  AA  C1  FA  C0  46
                                --
:01F8   E1  E9  A7  A7  79  A6  9C  E3
```

Die Speicherstelle, auf die der Stapelzeiger weist, ist unterstrichen. Nun starten wir mit G 1400 unser kleines verbotenes Testprogramm. Es meldet sich die Registeranzeige. Im Stapelzeiger steht jetzt F4 (oder eben Ihr vorhergegangener SP minus 2). Wenn wir nun wieder im Stapel nachsehen mit M 01F0 01FF, dann finden wir im Gegensatz zur obigen Anuzeige nun:

```
:01F0   20  AA  C1  FA  C0  02  14  46
                        --  ↑↑  ↑↑
:01F8   E1  E9  A7  A7  79  A6  9C  E3
```

Unterstrichen ist wieder das Ziel des Stapelzeigers, der jetzt zwei Plätze weitergerückt ist, um der durch Pfeile gekennzeichneten Adresse 1402 (als LSB/MSB) Raum zu schaffen. $1402 ist das letzte Byte des JSR-Befehls. Wie wir den Programmzähler kennen, ist er im allgemeinen immer einen Schritt voraus. Hier liegt er aber einen zurück, falls er nach Beendigung des Unterprogrammes an der notierten Adresse weitermacht. Dazu kommen wir gleich noch. Was wir am Programmzähler aber auch noch nach Ablauf unseres kurzen Beispielprogrammes ablesen können, ist die Tatsache, daß die Sprungadresse 1500 in ihn geschrieben wird, somit der Sprung dann also stattgefunden hat.

Nun bauen wir das kleine Programmchen etwas um:

```
1400   JSR 1500
1403   BRK
```

Das Unterprogramm soll nur aus dem Rücksprung bestehen:

```
1500   RTS
```

Verlangen Sie nun noch vor dem Start eine Registeranzeige mit R und merken Sie sich den Wert des Stapelzeigers. Dann starten Sie das Programm mit G 1400 und achten Sie auf die neue Registeranzeige. Zwei Dinge interessieren uns:

1) Der Wert des Stapelzeigers ist unverändert geblieben.
2) Der Programmzähler weist nun auf $1404.

Wenn Sie nun nochmal mit dem M-Befehl des SMON in den Stapel sehen, werden Sie unter Umständen zwar noch die Adresse 1402 dort finden (dann nämlich, wenn wir den Stapel seit dem letzten Programm nicht verändert haben). Wie Sie aber inzwischen wissen, hätte durch den neuen JSR-Befehl nochmal 1402 dort eingetragen sein müssen. Das stand da auch einige Mikrosekunden lang... bis der RTS-Befehl wirksam wurde. RTS macht ziemlich viel:

1) RTS holt die auf dem Stapel gespeicherte Adresse ab, und schreibt sie in den Programmzähler.
2) RTS vermindert dabei den Stapelzeiger um 2.
3) RTS addiert zum Programmzähler eine 1.

Deswegen kann das Programm also bei $1403 weiterlaufen und der Programmzähler nun hinter dem BRK-Befehl stehen.

Machen Sie doch mal etwas anscheinend total Verrücktes: Starten Sie mit G 1500. Es gibt da zwei Möglichkeiten, was geschehen kann: Entweder stand da noch vom ersten unterbrochenen Testprogramm die Adresse 1402. Dann endete nun alles mit einer Registeranzeige, bei der der Stapelzeiger um 2 höher gerutscht ist.

Oder da stand diese Adresse nicht mehr. Dann befinden Sie sich nun wieder im Basic. Wieso eigentlich? Als nächste Adresse finden Sie auf dem Stapel $E146 (dez.57670). Diese Adresse + 1 wird ja durch RTS in den Programmzähler gerufen. Ein Sprung an diese Adresse ist ein Sprung in ein Programm des Betriebssystems. Haben Sie ein ROM-Listing? Dann sehen Sie mal nach: Dort steht der Befehl...RTS. Dieses neuerliche RTS holt nun jedenfalls die nächste Adresse vom Stapel: $A7E9 (dez.42985). Diese Adresse + 1 im Programmzähler führt unseren Computer in die Basic-Interpreter-Schleife, also ins Basic zurück.

Wir haben so viel über den Stapel gehört, daß wir JSR fast schon wieder aus den Augen verloren haben. Deswegen nochmal eine kurze Übersicht:

a) JSR speichert den Programmzählerwert des letzten Bytes des Befehls auf dem Stapel
zum Beispiel 1402
b) stellt dabei den Stapelzähler um 2 zurück
zum Beispiel von $F6 nach $F4
c) schreibt in den Programmzähler die angegebene Zieladresse,
zum Beispiel 1500
d) Das Unterprogramm wird abgearbeitet bis der RTS-Befehl auftaucht.
e) Dann wird die gemerkte Adresse +1 in den Programmzähler geschrieben
zum Beispiel 1402+1=1403
f) und dabei der Stapelzähler wieder um 2 erhöht,
zum Beispiel von $F4 wieder zu $F6
g) Das Programm läuft nun wieder nach dem JSR-Befehl weiter, zum Beispiel also bei 1403.

Nun sollte eigentlich auch klar sein, warum ein Aussprung aus einem Unterprogramm oder ein Abbruch im Unterprogramm eine Programmierer-Todsünde ist: Der Stapelzeiger wird nicht zurückgestellt. Die gemerkte Rücksprungadresse versauert allmählich auf dem Stapel. Noch schlimmer sind solche Sachen in einer Schleife, wo mehrfach aus dem Unterprogramm ausgebrochen wird: Hier ist der Stapel bald voll Müll und der Computer beendet seine Zusammenarbeit mit dem Programmierer. Weil aber Basic-Programme nichts anderes sind als eine Folge von Maschinenprogrammen, die je nach Befehl durch den Interpreter aneinandergereiht werden, ist das auch in Basic eine Todsünde. Wir wollen aber nicht so hart mit uns umgehen: Wenn wir gelernt haben, wie man mit speziellen Assembler-Befehlen im Stapel herumschaufeln kann, dann haben wir bei richtiger Anwendung von vorneherein jedenfalls in diesem Punkt die Absolution erhalten.

Alles fließt: Fließkommazahlen

Jeder, der tiefer in die Geheimnisse der Assembler-Alchimie eindringen will, muß sich vertraut machen mit der häufigsten Art der Zahlenverarbeitung in unserem Computer. Das ist die Handhabung von Fließkommazahlen (auch Gleitkommazahlen genannt). Wir werden dazu folgende Fragen zu klären haben:

1) Was sind Fließkommazahlen?
2) Wie sehen sie im binären Zahlensystem aus?
3) Wie behandelt unser Computer positive und negative Fließkommazahlen?
4) Wie können wir als Programmierer Einfluß nehmen auf die Verarbeitung dieser Zahlen im Computer?

Die Behandlung dieser vier Fragen wird uns eine ganze Weile beschäftigen. Fangen wir mit der ersten an: In Standardwerken der Mathematik werden Sie lange suchen müssen, um den Begriff »Fließkommazahl« zu finden. Im deutschen Sprachraum gibt es häufiger die Bezeichnung »wissenschaftliche Zahlendarstellung«. Das klingt sehr hochgestochen und ist eigentlich ganz einfach. Leser der Grafik-Serie werden sich vielleicht noch erinnern: Die Zahl 1000 kann man auf verschiedene Weise darstellen:

1000 = 10 * 10 * 10 = 1013 ect.

Die hochgestellte Zahl (in Computerschreibweise: Die Zahl hinter dem Hochpfeil) ist hier gleich der Anzahl der Stellen minus 1 (1000 hat vier Stellen, also ist die Hochzahl eine 3). Diese Hochzahl nennt man Exponent (vom lateinischen exponere = anzeigen, herausheben). Nehmen wir nun einige andere Zahlen:

200 = 2 * 100 = 2 * 1012
oder
2500 = 2,5 * 1000 = 2,5 * 1013

Ich glaube, jetzt beginnt es Ihnen klarzuwerden, daß man auf diese Art wohl alle Zahlen irgendwie darstellen kann. Man dröselt die Zahlen auseinander, bildet ein Produkt, von dem der eine Multiplikator durch 10 teilbar ist (durch dio Basis unseres normalen Zahlensystems). Genauer gesagt: Ein Faktor (also in den Beispielen 1000 oder 100) ist darstellbar als Potenz von 10. Der andere Faktor (in den Beispielen 1 oder 2 oder 2,5) wird Mantisse (vom lateinischen manitissa = Zugabe, Anhang, Schleppe) genannt. Sehen wir uns nochmal 2500 an:

2500	=	2,5 * 1000	=	2,5 * 1013
	=	25 * 100	=	25 * 1012
	=	250 * 10	=	250 * 1011
	=	2500 * 1	=	2500 * 1010

Das letzte war nur der Vollständigkeit halber, denn irgendeine Zahl hoch 0 ist immer 1. Man kann auch aus der 2500 folgendes machen:

2500 = 0,25 * 10000 = 0,25 * 10↑4
oder = 0,025 * 100000 = 0,025 * 10↑5

und so weiter. Oder anders herum:

2500 = 25000 * 0,1 = 25000 * 10↑−1
= 250000 * 0,01 = 250000 * 10↑−2

und so weiter.

Dabei bedeutet:
10↑−2 = 1/10↑2 = 0,01

Man kann sich das merken, indem man die Anzahl der Stellen zählt, um die man das Komma verschiebt. Diese Anzahl addiert man dann zur Hochzahl. Zur Erläuterung:
0,12345 = 1,2345 * 10↑−1

Wir haben das Komma um eine Stelle nach rechts gerückt, weshalb wir die Hochzahl −1 schreiben müssen (vorher war da nämlich unsichtbar die Hochzahl 0: und 10↑0 = 1).
0,12345 = 123,45 * 10↑−3

Hier wurde das Komma um drei Stellen nach rechts verschoben. Daher der Exponent −3. Sie sehen folgenden Zusammenhang:

Komma eine Stelle nach rechts verschoben: Exponent + (−1).
Zum Beispiel
0,1234*10↑−2 = 1,234*10↑−3

Komma eine Stelle nach links verschoben: Exponent + 1.
Zum Beispiel
3,14*10↑2 = 0,314*10↑3

Verstehen Sie nun, warum man diese Art der Zahlendarstellung Fließkomma- oder Gleitkommazahlen nennt?

Vielleicht sehen Sie aber noch nicht den Sinn der Fließkommazahlen ein. Dazu gebe ich Ihnen zwei einsichtige Beispiele. Der Atomkern eines Heliumatoms wiegt etwa (halten Sie sich fest):
0,000 000 000 000 000 000 000 000 006 643 kg.

Sehr unbequem, diese ganzen Nullen immer mitzuschleppen. Wir verschieben deshalb das Komma um 27 Stellen nach rechts und schreiben dann
6,643*10↑−27 kg.

2. Beispiel: Wir haben einen Ballon mit diesem Gas gefüllt. Bei normalen Temperatur- und Luftdruckbedingungen befinden sich in einem Kubikzentimeter im Ballon ungefähr (nochmal festhalten!):
26 900 000 000 000 000 000 Heliumatome

Wieder eine recht unangenehme Nullschlepperei. Wir verschieben das Komma um 19 Stellen nach links und erhalten 2,69*10↑19 Heliumatome. Fein, nicht wahr!

Abgesehen von der höheren Bequemlichkeit: Der Computer müßte allerhand Speicherplatz zur Handhabung der vielen Nullen bereitstellen. Mit BCD-Zahlen könnten wir zwar jede Zahl erfassen, hätten aber immer unterschiedlich viele Bytes zu verarbeiten. Wenn wir Fließkommazahlen verwenden, können wir — wie Sie noch sehen werden — jede (na sagen wir mal: fast jede) Zahl in der gleichen Anzahl Bytes aufbewahren.

Vom Basic her kennen Sie Fließkommazahlen auch (hier wird das Komma allerdings durch den Punkt ersetzt, entsprechend der angloamerikanischen Schreibweise). Das sind die, wo man zum Beispiel schreibt 6.02E23 oder 6.02E+23, was dann bedeutet 6,02*10↑23. E steht dort für Zehnerexponent. Durch die Art, wie Fließkommazahlen im normalen Computerdasein gespeichert werden, ergeben sich obere und untere Grenzen. Die höchste in Basic verarbeitbare Zahl im C 64 ist
+1.70141183*10E38

Größere Zahlen verursachen in Basic einen OVERFLOW ERROR. Was in Maschinensprache mit größeren Zahlen geschieht, ist weitgehend unsere Sache. Die dem Betrag nach kleinste verarbeitbare Zahl ist
± 2.93873588*10E−39

In Basic arbeitet bei Unterschreitung der Computer einfach mit einer Null weiter. Für die Behandlung in Maschinensprache sind ebenfalls wir als Programmierer verantwortlich.

Für diesmal sei's genug der Zahlenspiele: In der nächsten Ausgabe werden wir uns weiter mit Fließkommazahlen befassen.

Die USR-Funktion

Wieder einmal soll uns das Zusammenspiel von Basic und Maschinensprache beschäftigen. Einen Aufruf von Maschinenroutinen — nämlich den mit SYS — haben wir schon kennengelernt. Wir POKEten die zu übergebenden Werte an die Abrufspeicherstellen. Bei diesen Werten hat es sich um einfache Integerzahlen gehandelt, zum Beispiel die Anzahl der Glieder einer zu summierenden arithmetischen Reihe. Was tun wir aber, wenn wir Fließkommavariable an ein Maschinenprogramm übermitteln wollen? Gewiß, werden Sie sagen, lernen wir das ja in den nächsten Folgen und können dann entsprechende POKE-Kommandos geben. Damit haben Sie auch recht, nur ist das dann der »harte« Weg. Es gibt auch einen problemlosen »weichen« Weg, nämlich das USR-Kommando.

USR ist ein Basic-Befehl und rührt her von »User callable machine language subroutine«, also »durch den Benutzer aufrufbares Maschinensprachunterprogramm«. Darin liegt eigentlich noch nichts Neues gegenüber dem SYS-Befehl. Im Gegensatz zu SYS — wo das Argument die Einsprungadresse des Maschinenprogrammes ist — übergibt USR als Argument eine beliebige Fließkommavariable in festgelegter Form an eine sehr nützliche Speicherstellenkombination, den Fließkomma-Akkumulator 1, von uns künftig einfach FAC genannt. Der FAC belegt die Speicherstellen 97 bis 102 ($61 bis $66). Wenn das eventuell in Basic benötigte Ergebnis dort auch in der vorgeschriebenen Form abgelegt wird, kann es im Basic-Programm weiterverwendet werden. Keine Angst, dazu kommen wir bei der weiteren Behandlung der Fließkommazahlen noch ganz ausführlich zu sprechen. Heute soll uns das noch nicht belasten. Als Argument kann man nämlich auch irgendeine bedeutungslose Größe, ein sogenanntes Dummy angeben, das dann gar nicht weiter verwendet wird. Der USR-Befehl dient in diesem Fall lediglich dem bequemen Ansteuern eines Maschinenprogrammes.

Woher weiß unser Computer beim USR-Befehl, welche Maschinenroutine er im 64-KByte-Speicher bearbeiten soll? Beim SYS-Befehl ist das klar: Das Argument sagt es:
SYS 24345
läßt den Programmzähler auf dez.24345 zeigen. Aber wenn wir eingeben:
USR(24345)
dann packt der Computer die Zahl 24345 als Fließkommavariable in den FAC und meldet dann einen SYNTAX ERROR. Das liegt daran, daß der Basic-Interpreter beim USR-Befehl einen der oben kennengelernten indirekten Sprünge vollführt:
JMP (311)

$311/312 (in dezimal 785/786) ist also ein Vektor, und der weist im Normalfall zu einer Routine, die den SYNTAX ERROR ausgibt (dez. 45640). Bevor wir also den USR-Befehl geben, müssen wir in diesen Vektor die Startadresse unserer Maschinenroutine schreiben:

dez. 24345 = $5F19
LSB $19 = dez. 25 in Speicher 785 mit POKE 785,25
MSB $5F = dez. 95 in Speicher 786 mit POKE 786,95

Jetzt weiß der Computer, wohin er beim USR-Aufruf springen soll, und solange, bis wir den Vektor wieder ändern, führt er bei jedem USR-Befehl unser bei 24345 stehendes Maschinenprogramm aus. Wir müssen nur noch dafür sorgen, daß dort dann auch wirklich eines anfängt. Ein Beispiel werden wir nachher noch behandeln.

Der harte Kern: Nochmal Speicherfragen

Die Struktur des C 64-Speichers ist vereinfacht schon in der Grafik-Serie und zu Beginn dieses Kurses gezeigt worden. Dabei tauchten zwei ROM-Bereiche auf, die wir Basic-Interpreter und Betriebssystem genannt haben. Diese Unterteilung ist nicht ganz korrekt. Wenn Sie über ein ROM-Listing verfügen und beispielsweise das Ende des ROM-Bereiches von $A000 bis $BFFF sowie den Anfang des oberen ROM ($E000 bis $FFFF) untersuchen, dann stellen Sie fest, daß ab dez. 49087 ($BFBF) die Basic-Funktion EXP bearbeitet wird. Der letzte Befehl vor $C000 beendet diese Funktion aber nicht etwa, sondern dort steht:
JMP E000

Tatsächlich läuft ab $E000 bis $E042 die Bearbeitung der EXP-Funktion munter weiter, und auch danach finden sich allerlei Basic-Befehle (SIN, COS und so weiter). Da liegt also keine klare Trennung vor, sondern ein Mischmasch. Wir sollten uns vielleicht angewöhnen — statt vom Interpreter und dem Betriebssystem —, vom unteren und oberen ROM-Bereich zu sprechen.

Eine andere Unterscheidung ist dagegen sinnvoll: Wie einige Besitzer neuerer Commodore 64 sicherlich bemerkt haben, sind Teile der ROM-Routinen im Laufe der Zeit verändert worden. Hauptsächlich geht es bei den aktuellen Neuerungen dieser internen Maschinenprogramme um die Farbgebung der Zeichen. Man kann eigentlich nie so recht wissen, was den Software-Planern von Commodore noch alles einfällt. Jedenfalls können deren Ideen manchmal recht dramatische Folgen haben, nämlich dann, wenn Sie ein fabelhaftes Maschinenprogramm gebaut haben, welches ROM-Routinen direkt verwendet. Der Programmierer spielt auf diese Weise eine milde Form des russischen Roulette. Glücklicherweise halten sich die Änderungen in Grenzen, und wir dokumentieren unsere Programme ja auch immer gut (Sie etwa nicht??). Notwendige Umbauten können also leicht vonstatten gehen.

Ganz ohne ROM-Routinen-Verwendung kommt man eigentlich kaum aus. Es gibt aber einen ROM-Bereich, für den Commodore verspricht, keinerlei Änderungen durchzuführen: die KERNAL-Sprungtabelle.

Das ist ein Programmbereich ($FF81 bis $FFF5), in dem 39 JMP-Befehle enthalten sind (zum Teil in absoluter, aber auch in indirekter Adressierung). Jeder dieser Sprungbefehle weist auf die Einsprungadresse eines Maschinenprogrammes. Da finden sich alle wichtigen Ein/Ausgabe-Operationen, Systemtakt- und Uhrsteuerungen und anderes mehr. Wir werden uns nach und nach damit vertraut ma-

chen. In der Tabelle 3 sind die KERNAL-Adressen und ihre Funktion aufgeführt. Manche davon können ohne jede Vorbereitung benutzt werden, andere brauchen bestimmte Routinen oder Angaben, um sinnvoll zu arbeiten.

Die Absicht von Commodore ist es, daß jeder Aufruf von zum Beispiel FFD2 die Ausgabe eines Zeichens bewirkt, und zwar unabhängig davon, welchen Computer in welcher Version wir benutzen. Das Programm, welches diese Zeichenausgabe letztendlich ausführt, kann sich ändern, kann in ganz andere Speicherbereiche gelegt werden. An der Stelle $FFD2 wird aber immer ein JMP mit der Einsprungadresse stehen. Leider ist diese Sprungtabelle viel zu knapp gehalten. Es gibt so viele interessante ROM-Routinen, die wir alle ohne diese schöne Sicherheit anspringen müssen.

Adresse HEX	dezimal	Name	Funktion
FF81	65409	CINT	Prüfen der TV-Norm, Berechnung der Taktfrequenz
FF84	65412	IOINIT	Ein/Ausgabe-Reset
FF87	65415	RAMTAS	Prüfen auf freien Basic-RAM
FF8A	65418	RESTOR	Initialisieren der I/O-Vektoren
FF8D	65412	VECTOR	Lesen und Setzen der I/O-Vektoren
FF90	65424	SETMSG	Setzen des Ausgabe-Modus
FF93	65427	SECOND	Ausgeben der Sekundäradresse nach LISTEN
FF96	65430	TKSA	Ausgabe der Sekundäradresse nach TALK
FF99	65433	MEMTOP	Lesen/Setzen des Speicherendes
FF9C	65436	MEMBOT	Lesen/Setzen des Speicheranfangs
FF9F	65439	SCNKEY	Abfragen der Tastatur
FFA2	65442	SETTMO	Setzen der Time-Out-Flagge
FFA5	65445	ACPTR	Zeichen vom seriellen Port in Akku lesen
FFA8	65448	CIOUT	Zeichen vom Akku auf seriellen Port ausgeben
FFAB	65451	UNTLK	Sendet UNTALK an seriellen Bus
FFAE	65454	UNLSN	Sendet UNLISTEN an seriellen Bus
FFB1	65457	LISTEN	Sendet LISTEN an Geräte per seriellen Bus
FFB4	65460	TALK	Sendet TALK an Geräte per seriellen Bus
FFB7	65463	READST	Liest I/O-Status in den Akku
FFBA	65466	SETLFS	Festlegung der Parameter für OPEN
FFBD	65469	SETNAM	Festlegung des Filenamens
FFC0	65472	OPEN	Öffnet spezifizierten File
FFC3	65475	CLOSE	Schließt spezifizierten File
FFC6	65478	CHKIN	Öffnet einen Eingabekanal
FFC9	65481	CHKOUT	Öffnet einen Ausgabekanal
FFCC	65484	CLRCHN	Schließt Ein- und Ausgabekanäle
FFCF	65487	CHRIN	Holt vom aktiven Eingabekanal ein Zeichen in den Akku
FFD2	65490	CHROUT	Sendet Akku-Inhalt auf aktiven Ausgabekanal
FFD5	65493	LOAD	LOAD und VERIFY von Programmen
FFD8	65496	SAVE	Speichern von Programmen
FFDB	65499	SETTIM	Uhrzeit setzen
FFDE	65502	RDTIM	Uhrzeit lesen
FFE1	65505	STOP	STOP-Taste abfragen
FFE4	65508	GETIN	Zeichen aus dem Tastaturpuffer in den Akku lesen
FFE7	65511	CLALL	Schließen aller Kanäle und Files
FFEA	65514	UDTIM	Uhr um 1/60 Sekunde weiterzählen
FFED	65517	SCREEN	Lesen des Bildschirmformates
FFF0	65520	PLOT	Lesen/Setzen der Cursor-Position
FFF3	65523	IOBASE	Lesen der Startadresse der Ein- und Ausgabebausteine

Tabelle 3. Kernal-Routinen

Die Urzelle eines Programmprojektes

Wir sind jetzt soweit, daß wir die Urzelle eines Programmprojektes, welches uns eine lange Zeit begleiten wird, aufbauen können. Wir wollen etwas unter den Teppich kehren. Der Teppich, das sind die uns bislang nicht zugängigen RAM-Bereiche unter den ROMs. Haben Sie das nicht auch schon mal erlebt, daß Sie während einer Programmarbeit plötzlich feststellen, Sie benötigen zum Beispiel für eine Zwischenrechnung ein weiteres Programm, oder Sie wälzen Listen und denken sich, ein kleiner Hilfsbildschirm wäre jetzt von Nutzen, oder....

Mit diesem heute zu startenden Programm wäre all das und noch viel mehr realisierbar. Es soll auf einfache Weise beliebige Speicherbereiche unters ROM schieben und sie wieder hervorholen können.

Natürlich braucht die Entwicklung dieses Projektes einige Zeit, zumal wir noch vieles lernen müssen. Deswegen sind wir in dieser ersten Urzelle noch sehr eingeschränkt: Wir verschieben zuerst einmal nur eine Bildschirm-Kopfzeile unter den oberen ROM-Bereich. Auch in dieser einfachsten Version gibt es noch einige Programmteile, die Sie erst nach der nächsten Ausgabe verstehen werden. Aber irgendwann müssen wir ja mal anfangen, Nägel mit Köpfen zu machen.

Unser Maschinenprogramm soll durch die USR-Funktion aufgerufen werden. Wie wir es in dieser Ausgabe gelernt haben, muß deshalb vor dem ersten Aufruf eine Initialisierung durch Belegen des USR-Vektors mit unserer Startadresse stattfinden.

Die Startadresse soll $02B6 (dez. 694) sein, denn dort gibt es einen freien RAM-Bereich bis inklusive $02FF (dez. 767), der weder andere Programme noch Kassettenoperationen stört. Das MSB $02 ist dezimal auch 2 und wird nach 786 gePOKEt:

```
POKE786,2
```

Das LSB $B6 ist dezimal 182 und soll in 785 geschrieben werden:

```
POKE785,182
```

Damit ist der USR-Vektor gestellt und wir brauchen uns nicht mehr weiter darum zu kümmern: Jeder USR-Aufruf wird nun den Start des Programmes bewirken. Nun zum Programm selbst. In Bild 1 finden Sie ein Flußdiagramm dazu.

Zunächst konstruieren wir den Teil, der die erste Bildschirmzeile nach $E000 und folgende Speicherstellen schiebt. Das X-Register verwenden wir als Index und laden es mit dez.40 = $27.

Schalten Sie also den SMON ein und starten Sie den Assembler mit:

```
A 02B6
```

Dann geben Sie ein:

```
02B6      LDX #27
```

Nun packen wir das letzte Zeichen der obersten Bildschirmzeile in den Akku:

```
02B8      LDA 0400,X
```

In das Y-Register legen wir die dazugehörige Farbe aus dem Bildschirmfarbspeicher:

```
02BB      LDY D800,X
```

Den Akkuinhalt — also die Bildschirminformation — legen wir nach $E000+$27:

```
02BE      STA E000,X
```

Dasselbe tun wir mit dem Farbcode, der ab $E028+$27 abwärts gespeichert wird. Leider kann man STY nicht X-indiziert-absolut adressieren (siehe Tabelle 1). Deshalb schieben wir zuerst den Y-Registerinhalt in den Akku:

```
02C1      TYA
02C2      STA E028,X
```

Damit ist das letzte Zeichen der Kopfzeile verschoben. Wir zählen das X-Register um 1 herunter:

```
02C5      DEX
```

Der X-Index weist nun auf das vorletzte Zeichen, mit dem sich alles ab $02A9 wiederholt. Wenn das X-Register bis 0 heruntergezählt ist, weist es auf das erste Zeichen der Kopfzeile. Die Schleife muß dann noch einmal durchlaufen werden und ein weiteres Herabzählen des X-Registers erzeugt $FF, was zum Setzen der N-Flagge führt. Das ist dann unser Signal, daß die gesamte Kopfzeile übertragen wurde. Die N-Flagge wird durch den BPL-Befehl getestet:

```
02C6      BPL 02B8
```

So weit, so gut. Wir hätten natürlich auch das X-Register von 0 an hochzählen können. Zum Beenden der Schleife wäre dann aber ein CPX-Befehl erforderlich gewesen, der jedesmal den X-Registerinhalt mit der Zahl $27 verglichen.

MERKE: Indexregister in Schleifen abwärts zu zählen, kann Rechenzeit einsparen!

Ab $02CE soll der umgekehrte Vorgang, also das Zurückschieben der vorher gespeicherten Kopfzeile in den Bildschirmspeicher geschehen. Das einfachste wäre es sicherlich, diesen Programmteil mit einem weiteren USR-Kommando zu starten. Das sähe dann so aus:

1.USR-Befehl – schiebt Kopfzeile unter oberes ROM
2.USR-Befehl – holt Kopfzeile zurück in Bildschirmspeicher
3.USR-Befehl – schiebt wieder Kopfzeile unter ROM
4.USR-Befehl – holt sie wieder zurück und so weiter.

Weil aber das Umstellen des USR-Vektors durch POKEs vom Basic aus lästig ist, tun wir dies einfach immer am Ende des betreffenden Maschinenprogrammabschnittes. Wir schreiben also das LSB der Programmfortführung ($CE) nach $311. Das MSB bleibt unverändert $02.

```
02C8      LDA #CE
02CA      STA 0311
02CD      RTS
```

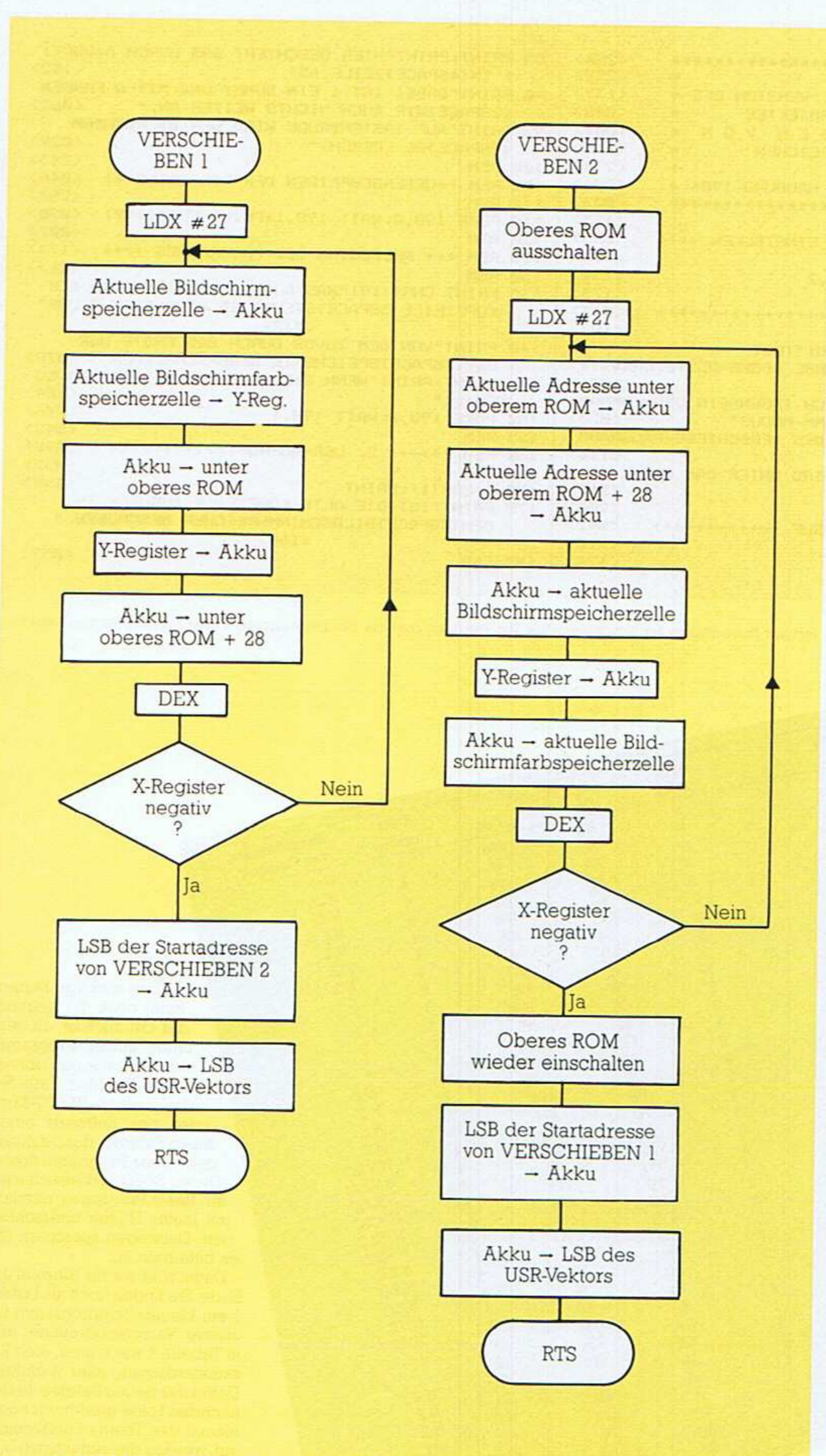


Bild 1. Das Flußdiagramm zu dem im Text erklärten Programm.

Mit dem RTS sind wir wieder im Basic-Programm gelandet, welches nun normal weiterverarbeitet wird. Erst ein neues USR-Kommando — im Programm oder im Direktmodus — startet den zweiten Teil unseres Maschinenprogrammes (weil in $0311, — der Einsprungpunkt des USR-Befehls — die Startadresse der auszuführenden Routine steht).

Einfache Befehle mit großer Wirkung

In diesem 2. Teil müssen wir erst einige Befehle geben, die Sie jetzt vielleicht noch nicht verstehen. Das hängt damit zusammen, daß zum Herauslesen des RAM unter dem ROM das ROM ausgeschaltet werden muß (entspricht POKE 1,53):

```
02CE    LDA 01
02D0    PHA
02D1    LDA #35
02D3    STA 01
```

(Der PHA-Befehl dient hier zur Zwischenspeicherung des Akku-Inhaltes). Das ist hiermit geschehen und wir kommen wieder in bekannte Gefilde mit der Ausleseschleife:

```
02D5    LDX #27
02D7    LDY E000,X
02DA    LDY E028,X
02DD    STA 0400,X
02E0    TYA
02E1    STA D800,X
02E4    DEX
02E5    BPL 02D7
```

Damit ist die gesamte gespeicherte Kopfzeile wieder zurückgeholt und wir können das ROM wieder einschalten:

```
02E7    PLA
02E8    STA 01
```

Falls nun wieder ein USR-Kommando auftaucht, soll die Kopfzeile mit dem 1. Programmteil unter das obere ROM gelegt werden wie am Anfang. Wir müssen deshalb den USR-Vektor auf $02B6 zurückschreiben:

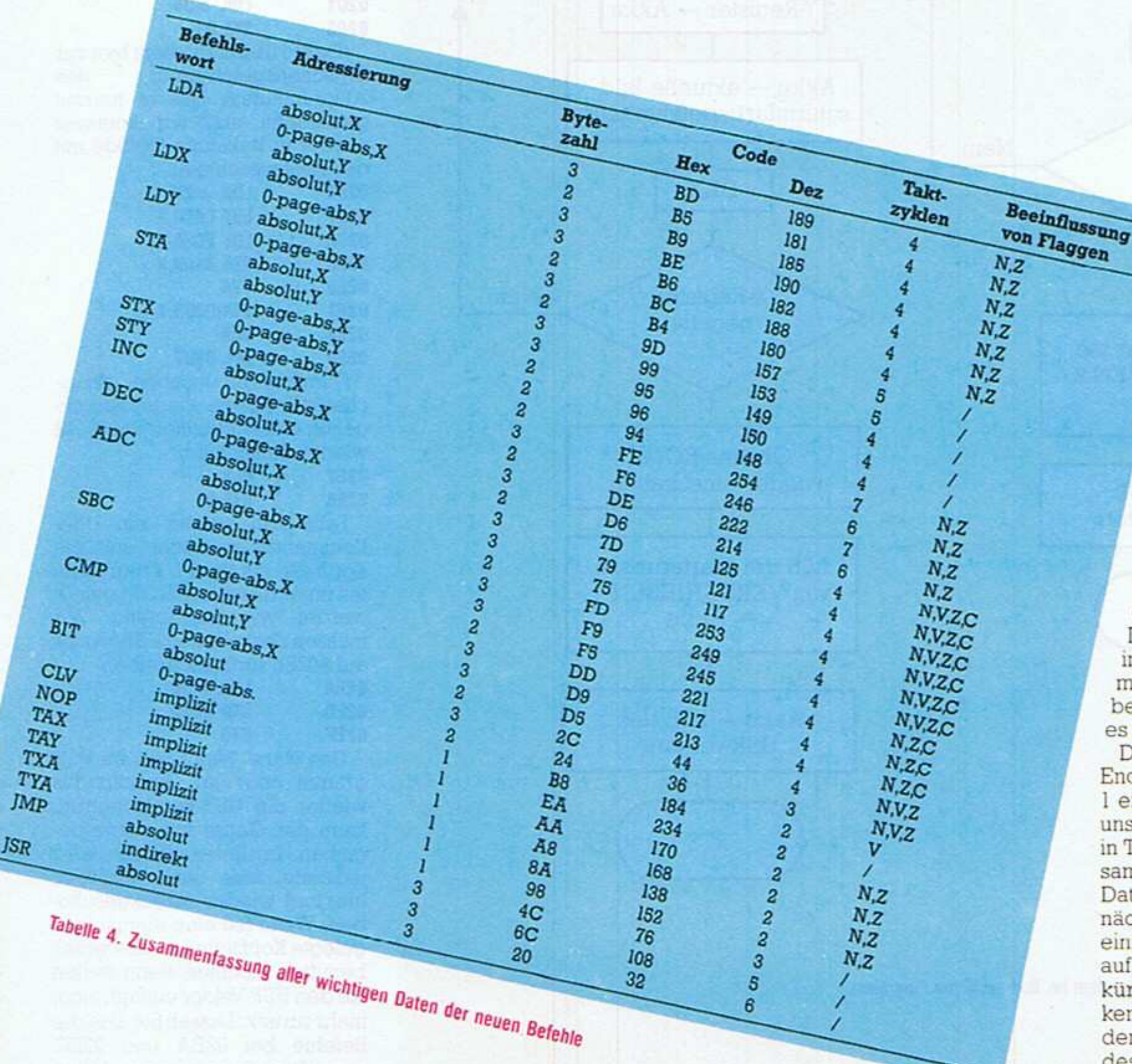
```
02EA    LDA #B6
02EC    STA 0311
02EF    RTS
```

Das wärs! Wenn nun im Programm oder im Direktmodus wieder ein USR-Befehl auftritt, kann das Ganze von vorne beginnen. In dieser Version wird jedesmal eine neue Kopfzeile hin- und wieder zurückgeschoben. Wenn Sie eine einmal festgelegte Kopfzeile immer wieder benutzen möchten, dann stellen Sie den USR-Vektor einfach nicht mehr zurück: Lassen Sie also die Befehle bei 02EA und 02EC weg. Das Programm endet in dem Fall mit:

```
02EA    RTS
```

```
1 REM *********************************                        <250>
2 REM *                               *                        <229>
3 REM *  TEST FUER DIE 1. VERSION DES *                        <139>
4 REM *        PROGRAMM-PROJEKTES     *                        <048>
5 REM * V E R S C H I E B E N  V O N  *                        <009>
6 REM *        SPEICHERBEREICHEN      *                        <193>
7 REM *                               *                        <234>
8 REM * HEIMO PONNATH    HAMBURG 1984 *                        <081>
9 REM *********************************                        <002>
10 REM                                                         <153>
15 REM ++++++ USR-VEKTOR EINSTELLEN ++++                       <065>
20 REM                                                         <163>
25 POKE 785,182:POKE 786,2                                     <239>
30 REM                                                         <173>
35 REM ++++++ KOPFZEILE ++++++++++++++++                       <013>
40 REM                                                         <183>
45 PRINT CHR$(147)CHR$(18)"TEST
   : BILD $0400=1024, FARBE $D800=55296"CHR$(14
   6)                                                          <071>
50 PRINT:PRINT:PRINT"DURCH IRGENDEIN USR-KOMMAN
   DO WIRD NUN IM PROGRAMM-MODUS"                              <020>
55 PRINT"DER ERSTE TEIL DES VERSCHIEBE-PROGRAMM
   ES AUFGERUFEN"                                              <110>
60 PRINT"DIE KOPFZEILE WIRD UNTER DAS OBERE
   ROM{2SPACE}KOPIERT."                                        <215>
65 REM                                                         <208>
70 REM ++++++ 1. USR-AUFRUF ++++++++++++                       <042>
75 REM                                                         <218>
80 A=USR(1)                                                    <124>
85 PRINT:PRINT"HIER GESCHIEHT DAS DURCH A=USR(1
   ) IN{4SPACE}ZEILE 65"                                       <132>
90 PRINT"DABEI IST 1 EIN DUMMY UND MIT A FANGEN
   {2SPACE}WIR AUCH NICHTS WEITER AN."                         <063>
95 PRINT"AUF TASTENDRUCK WIRD DER BILDSCHIRM
   {2SPACE}GE-LOESCHT"                                         <029>
100 REM                                                        <243>
105 REM ++UEBERSCHREIBEN DER KOPFZEILE ++                      <046>
110 REM                                                        <253>
115 POKE 198,0:WAIT 198,1:PRINT CHR$(147)                      <090>
120 REM                                                        <007>
125 REM +++ NEUBEGINN DES PROGRAMMES ++++                      <173>
130 REM                                                        <017>
135 PRINT CHR$(19)"WAS AUCH IMMER JETZT IN DER
    KOPFZEILE{3SPACE}STEHT, ES WIRD BEIM 2.USR"
    <017>
140 PRINT"VON DEM ZUVOR DURCH DAS ERSTE USR
    GE-{3SPACE}SPEICHERTE UEBERSCHRIEBEN"                      <078>
145 PRINT:PRINT"WENN SIE JETZT EINE TASTE DRUEC
    KEN..."                                                    <104>
150 POKE 198,0:WAIT 198,1                                      <246>
155 REM                                                        <042>
160 REM ++++++ 2. USR-AUFRUF +++++++++++                       <090>
165 REM                                                        <052>
170 A=USR(1):PRINT                                             <169>
175 PRINT"IST DIE ALTE KOPFZEILE ZURUECK IN
    DEN{3SPACE}BILDSCHIRMSPEICHER GESCHOBEN."
    <164>
180 END                                                        <052>
```

Listing 1. Test und Demonstration der Verschieberoutine. Das Programm zeigt das Ein- und Ausschalten einer Kopfzeile auf dem Bildschirm



Befehls-wort	Adressierung	Byte-zahl	Code Hex	Code Dez	Takt-zyklen	Beeinflussung von Flaggen
LDA	absolut,X	3	BD	189	4	N,Z
	0-page-abs,X	2	B5	181	4	N,Z
	absolut,Y	3	B9	185	4	N,Z
LDX	absolut,Y	3	BE	190	4	N,Z
	0-page-abs,Y	2	B6	182	4	N,Z
LDY	absolut,X	3	BC	188	4	N,Z
	0-page-abs,X	2	B4	180	4	N,Z
STA	absolut,X	3	9D	157	5	/
	absolut,Y	3	99	153	5	/
	0-page-abs,X	2	95	149	4	/
STX	0-page-abs,Y	2	96	150	4	/
STY	0-page-abs,X	2	94	148	4	/
INC	absolut,X	3	FE	254	7	N,Z
	0-page-abs,X	2	F6	246	6	N,Z
DEC	absolut,X	3	DE	222	7	N,Z
	0-page-abs,X	2	D6	214	6	N,Z
ADC	absolut,X	3	7D	125	4	N,V,Z,C
	absolut,Y	3	79	121	4	N,V,Z,C
	0-page-abs,X	2	75	117	4	N,V,Z,C
SBC	absolut,X	3	FD	253	4	N,V,Z,C
	absolut,Y	3	F9	249	4	N,V,Z,C
	0-page-abs,X	2	F5	245	4	N,V,Z,C
CMP	absolut,X	3	DD	221	4	N,Z,C
	absolut,Y	3	D9	217	4	N,Z,C
	0-page-abs,X	2	D5	213	4	N,Z,C
BIT	absolut	3	2C	44	4	N,V,Z
	0-page-abs.	2	24	36	3	N,V,Z
CLV	implizit	1	B8	184	2	V
NOP	implizit	1	EA	234	2	/
TAX	implizit	1	AA	170	2	N,Z
TAY	implizit	1	A8	168	2	N,Z
TXA	implizit	1	8A	138	2	N,Z
TYA	implizit	1	98	152	2	N,Z
JMP	absolut	3	4C	76	3	/
	indirekt	3	6C	108	5	/
JSR	absolut	3	20	32	6	/

Tabelle 4. Zusammenfassung aller wichtigen Daten der neuen Befehle

Eine wichtige Bemerkung noch: So bequem der Ort auch ist, an dem unser kurzes Programm steht, er hat einen gravierenden Nachteil: Falls Sie mittels einer RESET-Taste oder per Software einen Basic-Kaltstart durchführen, geht unser Programm flöten! Dieser Speicherbereich wird im Reset-Programm nämlich mit lauter Nullen überschrieben. Deswegen speichern Sie es bitte bald ab.

Damit sind wir für diesmal am Ende. Sie finden noch als Listing 1 ein kleines Testprogramm für unsere Verschieberoutine, und in Tabelle 4 wie immer, eine Zusammenfassung aller wichtigen Daten der neuen Befehle. In der nächsten Folge greifen wir nocheinmal das Thema Fließkomma auf, werden die einfachsten und kürzesten Kurzspeicher-Befehle kennenlernen und beginnen mit den leistungsfähigsten Befehlen des 6502, den indirekt-indizierten-Befehlen.

(H. Ponnath/gk)

Der gläserne VC 20

Teil 5

In Folge 4 haben wir die Grundlagen für das Arbeiten mit selbstdefinierten Grafikzeichen besprochen. Diesmal werden wir sehen, welche Fähigkeiten der VIC in bezug auf Grafik sonst noch besitzt.

Beginnen wollen wir heute mit dem Bitmapping, also dem Arbeiten mit hochauflösender Grafik. Hochauflösend deshalb, weil die einzelnen Grafikpunkte sehr klein und das Gesamtbild dadurch sehr fein (eben hochauflösend) ist. Vorher aber noch eine Begriffserklärung: Bitmapping (engl. Map = Landkarte) bedeutet etwa soviel, wie den Bildschirm zu kartografieren, das heißt jede Bildschirmposition ist in hochauflösender Grafik über eine Koordinate erreichbar. Wer das Superexpander-Modul (VC 1211A) besitzt, der kann mittels einfacher Befehle wie PLOT, DRAW, CIRCLE etc. mit der Hires-(high resolution) Grafik arbeiten. Beim »nackten« VC 20 ist dies jedoch nicht so einfach möglich.

Das Bitmapping

Im Gegensatz zu anderen Computern unterstützt der VC 20 diese Art Grafik überhaupt nicht; man muß sich also eine Softwarelösung einfallen lassen.

Die einzige Möglichkeit, einzelne Grafikpunkte (auch Pixels genannt) — aus denen sich ja jedes Zeichen zusammensetzt — anzusprechen, haben wir in der letzten Folge kennengelernt. Ich spreche von der Möglichkeit, sich Zeichen selbst zu definieren.

Wollen wir also Bitmapping betreiben, so bleibt uns nichts anderes übrig, als den gesamten Bildschirm mit verschiedenen Sonderzeichen vollzuschreiben und diese dann umzudefinieren, so daß ein komplettes, neues Bild in Hires-Grafik entsteht.

Doch mit diesem Vorhaben stößt man bereits auf erste Schwierigkeiten, denn der VC 20 kann ja nur 256 verschiedene Zeichen auf einmal auf dem Bildschirm darstellen. Daher müßte man sich mit einer relativ kleinen Fläche für die hochauflösenden Pixels zufrieden geben.

Da die Anzahl der darstellbaren Zeichen (diese setzen sich — um dies noch einmal zu wiederholen — aus 8 x 8 Pixels zusammen) auf 256 beschränkt ist, muß man sich eine andere Lösung einfallen lassen. Diese ist aber — man wird es kaum glauben — bereits in den VIC eingebaut worden. Das sieht praktisch so aus: Man vergrößert die im Bildschirmspeicher abgelegten Zeichen von 8 x 8 auf 16 x 8 Pixels (bei gleicher Auflösung), wodurch sich gleichzeitig die zur Verfügung stehende Zeichenfläche erhöht. Diese Vergrößerung wird über ein bestimmtes Bit im VIC-Kontrollregister #3 eingestellt (vergleiche Folge 4, Tabelle 3). Ist Bit 0 dieser Speicherstelle auf 0, so bleibt alles wie es war, das heißt jedes Zeichen wird innerhalb einer 8 x 8-Matrix dargestellt.

Setzt man dieses Bit nun aber mit »POKE 36867, PEEK (36867) OR 1« auf 1, so sind alle Zeichen plötzlich doppelt so hoch. Ein Charakter wird nämlich innerhalb eines 16 x 8-Gitters abgebildet (Bild 1). Das ist nun alles schön und gut, einen Nachteil hat dieser Betriebszustand aber (wer es selbst ausprobiert hat wird es sicherlich schon bemerkt haben). Denn mit dem normalen Zeichensatz kommt auf dem Bildschirm keine vernünftige Zeichenfolge mehr zustande. So wird beispielsweise aus dem @ (Klammeraffe) das Zeichen

@
A

aus dem A das Zeichen

B
C

und so fort. Drückt man nun eine Taste, so wird nicht der entsprechende Buchstabe abgebildet, sondern irgendwelche anderen Zeichen, die, wie eben beschrieben, übereinander gestapelt sind.

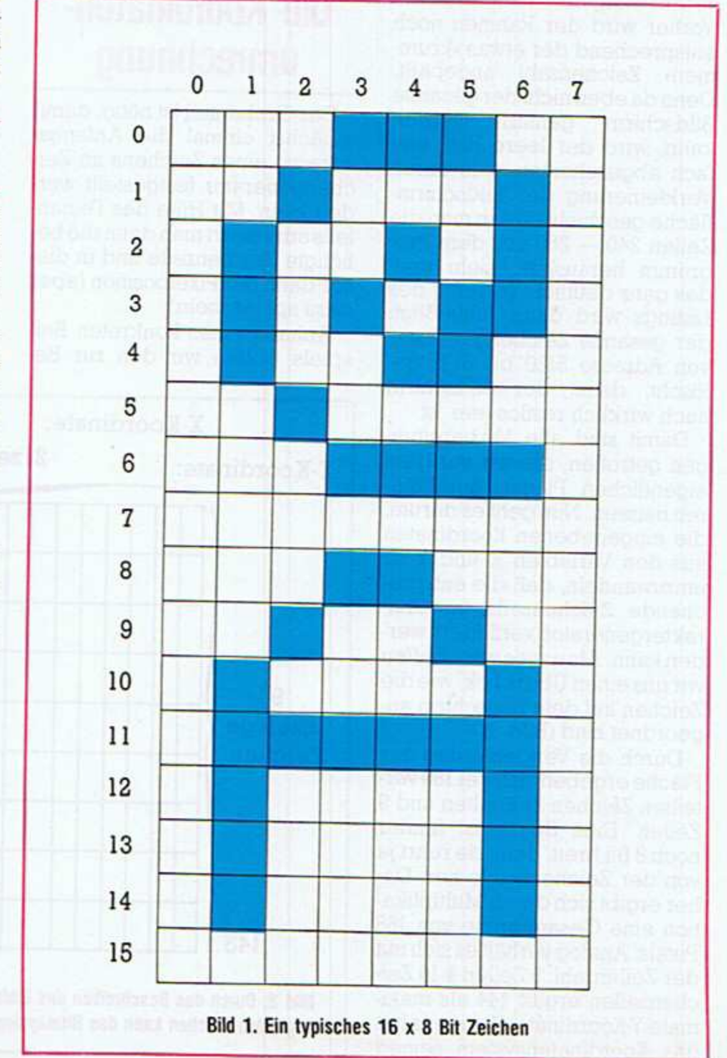


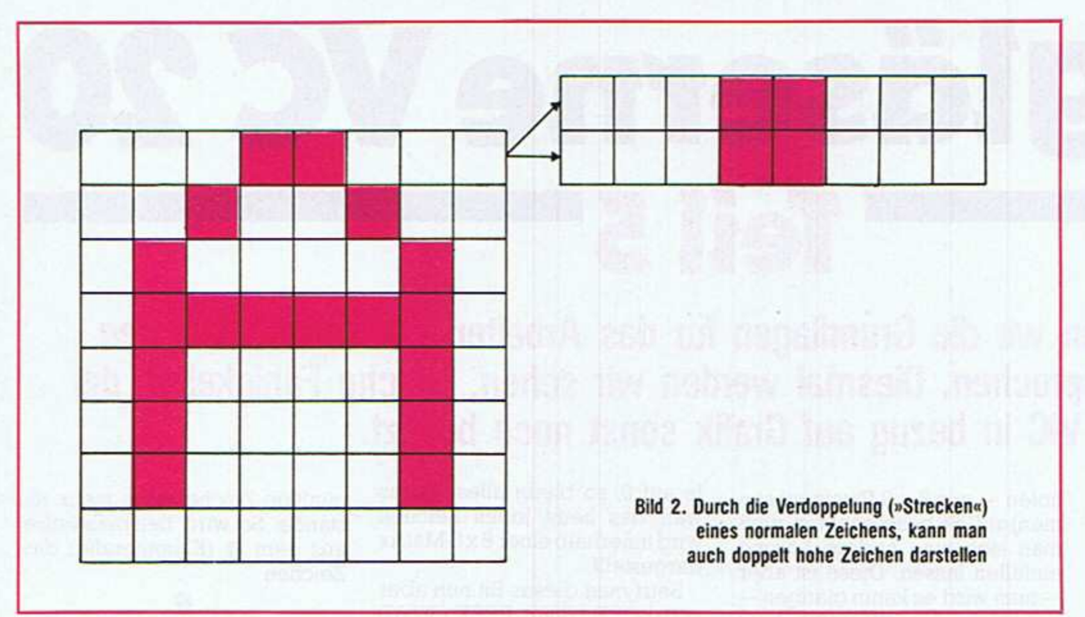
Bild 1. Ein typisches 16 x 8 Bit Zeichen

Doppelt hohe Zeichendarstellung

Die Erklärung dafür ist im Prinzip ganz einfach. Wie wir das letzte Mal gesehen haben, errechnet sich der VIC die relative Adresse eines Charakters im Zeichengenerator (relativ deshalb, weil die Adreßangaben auf eine Anfangsadresse bezogen sind), indem er den Bildschirmcode (auch hier erinnern wir uns daran, daß der Bildschirmcode die Reihenfolge der Zeichen im Charaktergenerator ist) jeweils mit 8 multipliziert.

Durch die Umschaltung auf 16-zeilige Zeichen liest der VIC für einen Charakter 16 Zeilen aus dem ROM. Daher werden — da das Zeichengenerator-ROM auf achtzeilige Zeichen ausgelegt ist — eben die Informationen von ursprünglich zwei Zeichen in einem dargestellt. Folglich multipliziert der VIC bei der Adreßermittlung den Bildschirmcode nicht mehr mit 8, sondern mit 16.

Auf diese Weise erklärt sich auch der Zahlensalat in diesem Betriebsmodus. Diese Darstellungsart hat jedoch den erheblichen Vorteil, daß jetzt mehr Zeichen als normalerweise abgebildet werden können. Es sind nämlich bereits in 128 Zeichen die Informationen von ursprünglich 256 Zeichen enthalten. Bei dem 129. Zeichen (RVS ON und @) beginnt daher schon der


Bild 2. Durch die Verdoppelung (»Strecken«) eines normalen Zeichens, kann man auch doppelt hohe Zeichen darstellen

zweite Zeichensatz des VC. Somit lassen sich diese zwei erstmalig gemeinsam auf dem Bildschirm ausdrucken. Statt bisher 256 »normaler« Zeichen läßt sich nun die gleiche Menge in doppelter Höhe (das entspricht also 512 regulären Zeichen) abbilden. Damit sind wir jetzt in der Lage, den gesamten Bildschirm füllen zu können. Darauf gehen wir aber später noch ausführlicher ein.

Das Naheliegendste wäre es ja nun, den Zeichengenerator ins RAM zu verlegen und dabei die Zeichen so umzubauen, daß man sie wieder richtig lesen kann. Dazu muß beim Kopieren jede Zeile verdoppelt werden; sonst bleibt ja alles beim Alten.

Das Programm in Listing 1 enthält zwei Zähler. Der eine zählt die ROM-, der andere die RAM-Speicherplätze, wobei letzterer genau doppelt so schnell läuft, denn eine ROM-Zeile soll ja zweimal hintereinander ins RAM geschrieben werden (dieses »Strecken« eines Zeichens ist in Bild 2 zu sehen). Da diese Schriftart natürlich besonders auffällig ist, eignet sie sich beispielsweise für Schaufensterwerbung oder ähnliches.

Wir wollen uns aber nun wieder unserem eigentlichen Thema — dem Bitmapping — zuwenden. Wie so oft stellt sich auch hier wieder die Frage nach dem Speicherplatz. Denn egal mit welcher Speicherausbauversion man gerade arbeitet, immer kommt es zu Kollisionen zwischen dem Zeichen- und dem Bildschirmspeicher. Daher müssen wir hier die Anzahl der verfügbaren Zeichen dementsprechend reduzieren. Wenn wir möglichst den ganzen Bildschirm füllen wollen, so muß es immer zu einem Kompromiß zwischen Bildschirmgröße und Speicherplatz kommen — logisch, denn je größer die verfügbare Hires-Fläche sein soll, um so mehr Speicherplatz benötigt man für die Sonderzeichen, über die das Bitmapping abgewickelt wird.

Ich glaube, mit 189 je 16 x 8 Bit-Zeichen einen solchen Kompromiß gefunden zu haben. Anhand von Listing 2, das in mehrere Teile gegliedert ist, möchte ich das Verfahren beim Bitmapping erklären. Programmteil 2 beschreibt den Bildschirm mit den veranschlagten 189 Zeichen. Vorher wird der Rahmen noch entsprechend der etwas »krummen« Zeichenzahl angepaßt. Denn da eben nicht der gesamte Bildschirm genutzt werden kann, wird der leere Rest einfach abgeschnitten, was durch Verkleinerung der Bildschirmfläche geschieht. Wenn man die Zeilen 240 — 260 aus dem Programm herausläßt, sieht man das ganz deutlich. In Teil 3 des Listings wird dann schließlich der gesamte Zeichengenerator von Adresse 5120 bis 8192 gelöscht, damit der Bildschirm auch wirklich restlos leer ist.

Damit sind alle Vorbereitungen getroffen, die wir vor dem eigentlichen Plotten durchführen müssen. Nun geht es darum, die eingegebenen Koordinaten aus den Variablen X und Y so umzuwandeln, daß die entsprechende Zeichenzeile im Charaktergenerator verändert werden kann. Als erstes verschaffen wir uns einen Überblick, wie die Zeichen auf dem Bildschirm angeordnet sind (Bild 3).

Durch die Verkleinerung der Fläche ergeben sich bei 189 verteilten Zeichen 21 Spalten und 9 Zeilen. Eine Spalte ist immer noch 8 Bit breit, denn sie rührt ja von der Zeichenbreite her. Daher ergibt sich durch Multiplikation eine Gesamtbreite von 168 Pixels. Analog verhält es sich mit der Zeilenzahl: 9 Zeilen à 16 Zeichenzeilen ergibt 144 als maximale Y-Koordinate. Übrigens hat das Koordinatensystem seinen Ursprung (X=0/ Y=0) nicht — wie in der Mathematik — links unten, sondern links oben.

Die Koordinaten müssen aus programmtechnischen Gründen in zwei Teile aufgespalten werden; nämlich in den sogenannten Grob- (oder auch Zeichen-) anteil und in den Feinanteil (auch Pixelposition genannt).

Die Koordinatenumrechnung

Der Grobanteil ist nötig, damit zunächst einmal die Anfangsadresse eines Zeichens im Zeichengenerator festgestellt werden kann. Mit Hilfe des Feinanteils adressiert man dann die benötigte Zeichenzeile und in dieser dann die Pixelposition (aber dazu später mehr).

Anhand eines konkreten Beispiels wollen wir den zur Berechnung nötigen Algorithmus entwickeln: Der Punkt mit den Koordinaten X= 43 und Y= 106 soll auf dem Bildschirm gesetzt werden.

Nun wird als erstes festgestellt, in welchem Zeichen eine Änderung vorgenommen werden muß. Zu diesem Zweck wird die Koordinate in die besagten Grobanteile aufgespalten, was durch einfache Division geschieht. Die Spaltenkoordinate wird durch 8 (Zeichenbreite), der Zeilenanteil wird durch 16 (Zeilenhöhe) dividiert:

X: 43 ÷ 8 = 5 Rest 3
Y: 106 ÷ 16 = 6 Rest 10

Das Ergebnis ist jeweils der Grobanteil, der Rest ist dann automatisch die Pixelposition (= Feinanteil). Der erste Teil dieser Rechnung wird in Listing 2 in den Zeilen 640 und 650 durchgeführt.

Danach wird die relative Position eines Zeichens im Charaktergenerator ermittelt. Die dafür nötigen »Formeln« haben wir ja bereits das letzte Mal besprochen:

Position = ZY x Zeichen pro Zeile + ZX
= 6 x 21 + 5 = 131.

Das betreffende Zeichen hat also den Bildschirmcode 131. Da ein Charakter (nach der Umschaltung auf eine 16 x 8 Matrix) den Platzbedarf von 16 Byte hat, kann man auch ganz leicht die Anfangsadresse der ersten Zeichenzeile errechnen:

ADRESSE = Position x Platzbedarf + Basisadresse = 131 x 16 + 5120 = 7216

Dann ermitteln wir als nächstes über den Pixelanteil der Y-Koordinate die benötigte Zeichenzeile. In unserem Beispiel muß zur Zeichenadresse der Wert von PY — also 10 — dazuaddiert werden. Damit haben wir die endgültige Adresse der an-

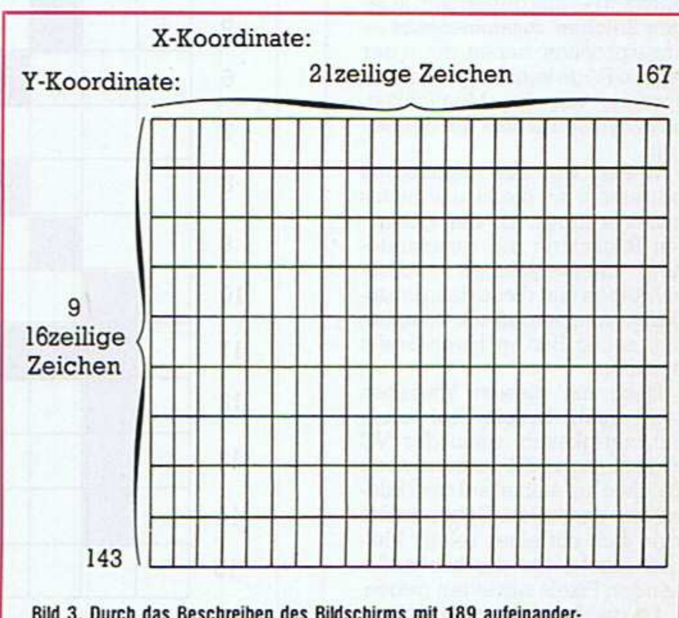


Bild 3. Durch das Beschreiben des Bildschirms mit 189 aufeinanderfolgenden Zeichen kann das Bitmapping realisiert werden

```
100 REM *****************************************  <009>
110 REM ***                                  ***   <249>
120 REM ***   DOPPELT HOHE ZEICHEN-          ***   <114>
130 REM ***   DARSTELLUNG.                   ***   <128>
140 REM ***                                  ***   <023>
150 REM ***     A C H T U N G  ! !           ***   <109>
160 REM ***   BEI 8 KBYTE ERWEITER.          ***   <071>
170 REM ***      VOR  DEM LADEN              ***   <102>
180 REM ***    >> POKE  44,32 <<             ***   <091>
190 REM ***       EINGEBEN !!!!              ***   <010>
200 REM *****************************************  <109>
210 POKE 55,0:POKE 56,24:CLR                      <211>
220 REM *** ZEILE 210 ENTFAELLT BEI               <100>
230 REM *** DER 8 KBYTE ERWEITERUNG.              <004>
240 AW=6144:EW=7147:G=14                          <252>
250 REM *** BEI 8 KBYTE ERWEITERUNG               <223>
260 REM *** ZEILE 230 FOLGENDERMASSEN             <125>
270 REM *** AENDERN :                             <083>
280 REM *** AW=5120:EW=8192:G=13                  <204>
290 Z1=32768                                      <106>
300 FOR Z2=AW TO EW STEP 2                        <159>
310 WE=PEEK(Z1)                                   <035>
320 POKE Z2,WE:POKE Z2+1,WE                       <044>
330 Z1=Z1+1:NEXT                                  <170>
340 POKE 36869,PEEK(36869)OR G                    <066>
350 POKE 36865,21                                 <145>
360 POKE 36867,33                                 <160>
```

Listing 1. Doppelt hohe Zeichendarstellung

gewählten Zeichenzeile berechnet. Dies klingt alles viel komplizierter, als es in Wirklichkeit ist, denn alle drei Schritte können zu einem (Zeile 690) zusammengefaßt werden.

Damit sind wir schon fast am Ziel (das kann man hier sogar wörtlich nehmen) angelangt. Als letztes muß das durch die Pixel-X-Koordinate vorgegebene Bit in der Zeichenzeile gesetzt werden. Dies wird durch eine ODER-Verknüpfung des Wertes mit der Zeichenzeile erreicht.

Vorher ist aber noch eine letzte Hürde zu überwinden, die Koordinate ist nämlich nicht Byte-identisch. Was bedeutet das? Nun, unser Bildschirm entspricht beim Bitmapping ja auch einem Koordinatensystem. Denn über die horizontale (X-) und die vertikale (Y-) Koordinate läßt sich jeder beliebige Pixelpunkt durch ein Zahlenpaar (eben durch den X- und Y-Wert) eindeutig adressieren. Dabei hat der Punkt links oben die Koordinaten (0/0), der rechts unten die Koordinaten (167/143).

Für die X-Achse bedeutet dies, daß der Wert nach rechts ansteigt. Für die Fein-X-Koordinate gilt natürlich das Gleiche; sie kann von links nach rechts folgende Positionen annehmen: 0,1,2,3,4,5,6,7.

Hier liegt nun der springende Punkt. Position 0 entspricht nämlich Bit 7 in der Zeichenzeile, Position 1 Bit 6, die Position 2 Bit 5 und so fort. Diese Bits laufen also genau in entgegengesetzter Richtung. Folglich muß die Pixel-X-Koordinate dementsprechend »umgepolt« werden. Dies wird ganz einfach dadurch erreicht, indem man diesen Wert von der 7 subtrahiert:

Bitformat = 7 — (Pixel-X-Koordinate)

In unserem Beispiel ergibt sich: Bitposition = 7 — 3 = 4

Bit 4 soll nun in der Zeichenzeile gesetzt werden. Dies erreichen wir — wie bereits erwähnt — durch die ODER-Verknüpfung des Wertes mit dem Zeichenbyte. Wer den letzten Teil aufmerksam verfolgt hat, dem wird dies nicht schwerfallen. Da Bit 4 die Wertigkeit 16 (= 2^4) hat, wird die Zeichenzeile eben mit 16 ODER-verknüpft.

Wenn man alle drei Schritte zusammenfaßt, ergibt sich folgende Zeile:

POKE AD, PEEK (AD) OR 2 ↑ (7-PX).

Damit ist der entsprechende Pixelpunkt gesetzt. Mit diesem letzten Teilstück haben wir nun endlich den kompletten Routinenteil beieinander, um ihn als Unterprogramm in Teil 4 von Listing 2 zu verwenden. Natürlich können mit Hilfe dieser Methode auch Punkte gelöscht oder abgefragt werden. Das Löschen wird mit Hilfe der AND-Operation bewerkstelligt. Das zu löschende Bit muß im Operanden auf Null, alle anderen, die unberührt bleiben sollen, auf eins gesetzt werden. Folgende Zeile löscht den adressierten Pixelpunkt:

POKE AD, PEEK (AD) AND 255—2 ↑ (7-PX).

Die Abfrage von Punkten wird ebenfalls über die AND-Operation abgewickelt. Das gewünschte Bit wird im Operanden gesetzt und danach mit der Speicherstelle UND-verknüpft. Ist das adressierte Pixel gesetzt, so ist die IF-THEN-Bedingung erfüllt, ansonsten nicht:

IF 2 ↑ (7-PX) = (PEEK(AD) AND 2 ↑ (7-PX)) THEN...

Natürlich sollte man anstelle des Ausdrucks 2 ↑ (7-PX) eine Variable definieren, damit das Programm kürzer und schneller wird.

Zeichnen auf dem Bildschirm — der Joypainter

Soweit also die Erklärung des Bitmapping beim VC 20. Um einmal zu zeigen, was man mit diesen Erkenntnissen anfangen kann, habe ich ein Joypainter-Programm in Maschinensprache entwickelt. Der Basic-Lader (Listing 3 und 4) transferiert das Programm aus den DATA-Zeilen automatisch in die Speicherbereiche ab $2000 und nur dort ist es lauffähig (Der Speicher muß also um mindestens 8 KByte erweitert sein). Beide Listings müssen nacheinander geladen und gestartet werden. Zunächst zur Bedienung der Routine, die mit »SYS 9682« gestartet wird.

Der Joystickpainter arbeitet — wie der Name bereits sagt — mit dem Joystick. Der kleine »Zeichencursor« kann mit dem Steuerknüppel in alle vier Himmelsrichtungen und in alle Diagonalen bewegt werden.

Es ist aber auch möglich, den Cursor über die Tasten E, S, D und X zu bewegen (ganz nach Belieben). Auch die Cursortasten können für die Steuerung herangezogen werden, über die Tastatur hat man allerdings nur vier Bewegungsrichtungen für den Zeichencursor zur Verfügung. Die Tasten verwendet man sinnvollerweise dann, wenn es darum geht, besonders exakt zu zeichnen. Aus diesem Grund kann man auch die Bewegungsgeschwindigkeit über die Funktionstasten F5 und F7 auf schnell beziehungsweise langsam stellen.

Gezeichnet wird mit den Funktionstasten F1 und F3. Die obere setzt einen Punkt (und rückt den Cursor um eins nach rechts), die andere löscht einen Punkt (diese Tasten haben eine Wiederholungsfunktion). Auch der Feuerknopf kann zum Zeichnen verwendet werden. Ein kurzer Druck bewirkt das Setzen, ein langer das Löschen eines Pixels. Über die CTRL-Taste können darüber hinaus noch andere Funktionen wie Dauerzeichnen oder Dauerlöschen angewählt werden. Tabelle 1 zeigt den kompletten Befehlsvorrat.

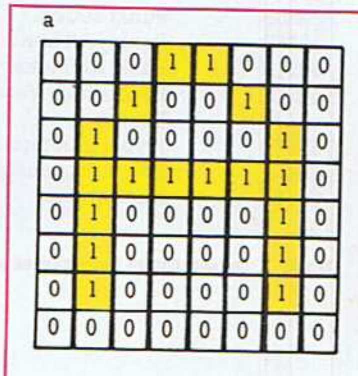
Natürlich fehlen diesem Programm — da es nicht allzu lang ist — einige Funktionen, die es noch komfortabler machen würden, wie beispielsweise die Verschiebung eines Zeichenblocks auf dem Bildschirm. Solch ein Programm würde aber samt Erklärung den Rahmen dieses Kurses sprengen.

Diejenigen, die einige Routinen wie das Bitmapping-Unterprogramm oder ähnliches in ihren eigenen Programmen verwenden möchten, finden in Tabelle 2 eine Auflistung der wichtigsten Programmteile.

Das Unterprogramm zur Joystickabfrage wird übrigens in der nächsten Folge ausführlicher besprochen. Einen bedeutenden Nachteil hat die Hires-Grafik aber, sie ist nämlich ziemlich farblos. Warum ist dies so?

Da sich diese Grafikart aus den 8 x 8- (oder 16 x 8-) Basiszei-

a



b

0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0

c

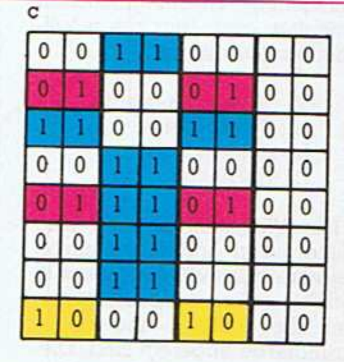


Bild 4. Das Hires-Zeichen »A« (4a) wird im Multicolormodus (4b) zu einem mehrfarbigen Gebilde. Der Vielfarbmodus eignet sich aber besonders für Spielfiguren (4c) oder ähnliches.

chen zusammensetzt, kann man — wie es auch bei »normalen« Buchstaben und Grafikzeichen üblich ist — jeder Videospeicherposition über das Farb-RAM jeweils nur eine Farbe (von 0 bis 7) zuweisen.

Die Darstellung von verschiedenen Farben innerhalb eines Zeichens ist also nicht möglich. Für manche Zwecke benötigt man aber nun gerade mehrere Farben innerhalb einer Zeichenmatrix. In diesem Fall wird dann ganz einfach auf den Multicolormodus — den wir im folgenden genauer besprechen werden — umgeschaltet.

Buntes Allerlei: Multicolor

Ist innerhalb eines 8 x 8 Zeichengitters irgendwo ein Bit mit dem Wert 1, so wird die entsprechende Bildschirmzelle mit dem Farbwert, der in der zugehörigen Farbspeicherstelle steht, ausgefüllt. Befindet sich in dieser Pixelzelle der Binärwert 0, so wird diese in der Hintergrundfarbe auf dem Bildschirm abgebildet (sie ist also nicht zu sehen).

In einem normalen hochauflösenden Zeichen können also zwei verschiedene Farben dargestellt werden:
Bit auf 0: Pixeldarstellung in Hintergrundfarbe
Bit auf 1: Pixeldarstellung in Vordergrundfarbe (Farb-RAM)

Wie Sie sehen, kann man die zwei Zustände auch als Farbcodes interpretieren. Ebenso verhält es sich im Multicolormodus: Hier werden nun zwei Bit in einer Zeichenzeile zusammengefaßt, die ebenfalls einen solchen Farbcode (allerdings für vier Farben) bilden. Ein Pixel ist in diesem Fall dann aber auch doppelt so breit wie normal, wodurch sich auch die Auflösung um die Hälfte reduziert. Bild 4 verdeutlicht den Unterschied zwischen diesen beiden Grafikarten. Diese vier möglichen Farbcodes sind auch hierbei wiederum nur die Zeiger auf Speicherstellen, in denen die eigentlichen Farbinformationen enthalten sind. Hier die möglichen Kombinationen:

00 : Hintergrundfarbe
01 : Rahmenfarbe
10 : Zeichenfarbe
11 : Hilfsfarbe

Die Werte 00 und 01, also Hintergrund- und Rahmenfarbe zeigen auf das VIC-Register #15 (36879), in dem ja die Informationen über Rahmen- und Hintergrundfarbe abgelegt sind. Diese Farben sind also für alle Multicolorzeichen einheitlich vorgewählt. Auch der Code 11 — die Hilfsfarbe — wird einheitlich vorgewählt. Dieser Farbwert wird allerdings in einem Register gespeichert, das eigentlich mit Grafik überhaupt nichts zu tun hat, nämlich dem VIC-Register # 14 (36878), mit dem ja auch die Lautstärke festgelegt wird. Die Bits 0 bis 3 enthalten also den Lautstärkewert, die oberen vier Bit die Hilfsfarbe. Diese muß demnach speziell für Multicolor eingestellt werden, denn normalerweise ist diese Speicherstelle auf Null gesetzt (damit die Lautstärke ebenfalls null ist).

Schließlich ist noch Farbcode 10 zu erwähnen. Dieser wird für jedes Zeichen einzeln in der entsprechenden Farbspeicherstelle — wie wir es ja von den Hires-Zeichen her kennen — bestimmt. Die Bits 0 bis 2 speichern die gewünschte Farbe, Bit 3 hat eine besondere Funktion: Über dieses Bit wird — und das ist neu — der Grafikmodus ausgewählt. Ist es auf 0 (was ja der Normalfall ist), so stellt der Computer den in der dazugehörigen Videospeicherposition abgelegten Charakter im Zweifarbmodus (also in Hires-Grafik) dar. Setzt man das Bit 3 nun aber auf 1, so wird der Vierfarbmodus für dieses Zeichen eingeschaltet.

Folglich kann also die Darstellungsart für jedes Zeichen im Farb-RAM selektiert werden. Daher kann also sowohl der eine, als auch der andere Grafiktyp gleichzeitig auf dem Bildschirm abgebildet werden.

Probieren Sie es gleich einmal aus: die 1, also das Zeichen A, wird in die Bildschirmspeicherstelle 7680 (4096) gePOKEt. Danach ist in der entsprechenden Farbspeicherstelle der Colorwert mit POKE 38400,6 (beziehungsweise 37888,6) zu vermerken. In der linken oberen Ecke steht also nun ein blaues »A«. Als nächstes wird Bit 3 im Farb-RAM für diese Zeichen gesetzt: POKE 38400, PEEK (38400) OR 8.

Nun verschwimmen die Konturen des Charakters und ein farbiges Gebilde erscheint, das nur noch entfernt an den Buchstaben A erinnert. Nun, wie Sie schon bei den doppelt hohen Zeichen gesehen haben, ist das Zeichengenerator-ROM für eine bestimmte Betriebsart konzipiert. Natürlich werden die Zeichen deshalb in hochauflösender Grafik dargestellt, damit sie besser unterscheidbar sind. Da der Zeichengenerator wiederum nur für Hires-Zeichen konzipiert wurde, kann man in Multicolor nicht viel mit ihnen anfangen.

Wegen ihrer geringen Auflösung eignet sich diese Betriebsart sowieso nur für Spielfiguren oder ähnliche Zeichen. Dazu muß der Zeichengenerator wieder ins RAM verlegt werden, was ja durch eine Änderung im Register #5 geschieht. Dann können die Zeichen wie in Bild 4 programmiert werden. In Listing 5 ist eine solche Routine zu finden. Sie erzeugt ein kleines Multicolor-Männchen (der Klammeraffe @ wird entsprechend umdefiniert).

Soweit unser etwas ausgedehnter Exkurs in die Welt der VC 20-Grafik, mit dem ich diese Folge beschließen möchte. In der nächsten Folge unseres Kurses möchte ich noch einmal auf das Betriebssystem und dessen Routinen eingehen, denn auch hier tut sich dem Programmierer ein weites Betätigungsfeld auf.

(Christoph Sauer/ev)

Taste	Funktion
F1	Punkt setzen
F3	Punkt löschen
F5	Cursor auf schnelle Bewegung umschalten
F7	Cursor auf langsame Bewegung umschalten
E	
S	
D	Steuertasten für den Zeichencursor
X	
CTRL C	Bildschirm löschen
CTRL F	Farbe des Bildschirmes mit den Tasten + und — ändern. Mit RETURN beendet man die Farbeinstellung
CTRL L	LOAD von Band oder Floppy
CTRL S	SAVE von Band oder Floppy
CTRL F1	Zeichenmodus: Der Joystick (oder die Tasten) werden zu einem Zeichenstift, das heißt, bei jeder Bewegung wird ein Punkt gesetzt.
CTRL F3	Radiermodus: Joypainter wird auf Dauerlöschen umgeschaltet.
SHIFT	»hebt« den Zeichenstift, beziehungsweise den Radiergummi. Solange man diese Taste drückt, kann man den Cursor bewegen, ohne daß das Bild verändert wird.
RUN/STOP	Hebt den Zeichen- beziehungsweise Radiermodus ganz auf

Tabelle 1. Die Steuerfunktionen beim Joypainter-Programm

Anfangsadresse	Funktion
$2000	Cursortasten abfragen
$2047	Joystickabfrage
$2084	Startbild (64'er....)
$21B2	Grafik (Bitmapping) einschalten
$21DC	Grafik löschen
$21F5	Koordinaten-Umrechnung
$225F	Punkt setzen
$226B	Punkt löschen
$2279	Punkt abfragen
$22D9	Feuerknopf Verarbeitung (kurzer Druck, langer Druck)
$233D	Joypaintroutine
$2410	Kontrolltastenabfrage und -verarbeitung
$25D2	Startadresse

Tabelle 2. Die wichtigsten Unterroutinen im Joypainter-Programm

```
100 REM ****************************           <009>
110 REM *** VC 20 BITMAPPINGROUTINE ***        <005>
120 REM *** FUER SPEICHER  >8 KBYTE ***        <125>
130 REM ***                         ***        <013>
140 REM ***    A C H T U N G !      ***        <066>
150 REM ***    VOR  DEM  LADEN      ***        <082>
160 REM ***   >> POKE  44,32 <<     ***        <071>
170 REM ***     EINGEBEN !!!!       ***        <246>
180 REM ****************************           <089>
190 REM                                        <077>
200 REM                                        <087>
210 REM ---------- TEIL 1 ------------         <203>
220 REM                                        <107>
230 REM *** BILDSCHIRM EINRICHTEN              <189>
240 POKE 36864,13  : REM LINKER RAND           <213>
250 POKE 36865,44  : REM OBERER RAND           <222>
260 POKE 36866,21  : REM 21 SPALTEN            <123>
270 POKE 36867,19  : REM 18 ZEILEN (16*8)      <093>
280 POKE 36869,205 : REM ZEICHEN AB 5120       <157>
290 REM                                        <178>
300 REM ---------- TEIL 2 ------------         <039>
310 REM                                        <198>
320 REM *** 189 ZEICHEN IN DEN BILD-           <172>
330 REM *** SCHIRMSPEICHER POKEN.              <028>
340 FOR T=0 TO 188:POKE 4096+T,T               <115>
350 POKE 37888+T,6:NEXT                        <036>
360 REM                                        <248>
370 REM ---------- TEIL 3 ------------         <110>
380 REM                                        <012>
390 REM *** HIRESSCREEN LOESCHEN               <032>
400 FOR T=0 TO 3024:POKE 5120+T,0:NEXT         <100>
410 REM                                        <042>
420 REM ---------- TEIL 4 ------------         <161>
430 REM                                        <062>
440 REM                                        <072>
450 REM *** HIER  WURDE  ALS  DEMON-           <255>
460 REM *** STRATIONSBEISPIEL  EINE            <015>
470 REM *** SIN-FUNKTION EINGESETZT.           <153>
480 FOR X=0 TO 167                             <222>
490 Y=(SIN(X*2*π/167)+1)*71.7                  <215>
500 GOSUB 640                                  <028>
510 NEXT                                       <129>
520 GOTO 520                                   <042>
530 REM                                        <163>
540 REM ---------- TEIL 5 ------------         <027>
550 REM                                        <183>
560 REM ***    BITMAPPINGROUTINE               <080>
570 REM ***    =================               <086>
580 REM *** DIE HOCHAUFLOESENDEN PIXELS        <149>
590 REM *** KOENNEN  UEBER  KOORDINATEN        <028>
600 REM *** ANGESPROCHEN WERDEN.               <215>
610 REM *** DIE X-KOORDINATE MUSS <168,        <007>
620 REM *** Y MUSS  <144 SEIN.                 <078>
630 REM                                        <007>
640 ZX=INT(X/8) : PX=X-ZX*8                    <086>
650 ZY=INT(Y/16): PY=Y-ZY*16                   <195>
660 REM *** DIE KOORDINATEN  WERDEN IN         <015>
670 REM *** EINEN ZEICHEN-  UND  EINEN         <165>
680 REM *** PIXELANTEIL AUFGESPALTEN.          <163>
690 AD=5120 +ZX*16 +ZY*336 +PY                 <026>
700 REM *** AD  IST  DIE ZEILENADRESSE         <224>
710 REM *** EINES  CHARAKTERS  IM ZEI-         <092>
720 REM *** CHENGENERAROR RAM.                 <176>
730 POKE AD,PEEK (AD) OR (2↑ (7-PX))           <120>
740 REM *** IN DIESER  ZEILE WIRD  DAS         <205>
750 REM *** GEWUENSCHTE PIXEL GESETZT.         <021>
760 RETURN                                     <136>
```

Listing 2. Die Bitmapping-Routine

```
100 REM *** JOYPAINT TEIL 1                    <062>
110 REM >> VOR DEM LADEN <<                    <034>
120 REM POKE 44,39:POKE9984,0                  <217>
130 REM >> EINGEBEN <<                         <066>
140 FOR T=8192 TO 9015:READ D:S=S+D:POKE T,D
  :NEXT                                        <048>
150 IF S<>92869 THEN PRINT"{CLR,3DOWN}FEHLER
   IN DATAS !!":END                            <234>
160 PRINT"BITTE TEIL 2 LADEN"                  <185>
170 DATA 095,234,152,072,165,197,162,003  <050>
180 DATA 221,059,032,240,005,202,016,248  <039>
190 DATA 208,005,189,063,032,208,027,162  <064>
200 DATA 001,221,067,032,240,005,202,016  <045>
210 DATA 248,208,005,189,069,032,208,004  <090>
220 DATA 169,000,240,006,172,141,002,208  <075>
230 DATA 001,010,133,184,104,168,234,234  <089>
240 DATA 165,184,096,041,018,026,049,004  <118>
250 DATA 008,002,001,023,031,004,001,120  <077>
260 DATA 152,072,169,000,141,019,145,169  <135>
270 DATA 127,141,034,145,173,032,145,041  <134>
280 DATA 128,073,128,074,074,074,074,133  <163>
290 DATA 184,169,255,141,034,145,173,017  <173>
300 DATA 145,041,060,073,060,074,074,133  <166>
310 DATA 185,041,008,074,074,074,133,250  <184>
320 DATA 165,185,041,007,005,184,170,104  <189>
330 DATA 168,138,088,096,169,059,141,015  <230>
340 DATA 144,162,000,189,160,032,032,210  <196>
350 DATA 255,232,208,247,189,160,033,240  <229>
360 DATA 006,032,210,255,232,208,245,096  <229>
370 DATA 147,031,206,163,163,205,032,032  <231>
380 DATA 032,206,163,165,032,207,165,013  <245>
390 DATA 165,207,208,186,032,032,206,206  <006>
400 DATA 165,165,032,204,165,013,165,165  <019>
410 DATA 032,032,032,206,206,032,165,165  <014>
420 DATA 032,032,032,206,163,163,205,167  <028>
430 DATA 208,206,208,013,165,163,163,205  <045>
440 DATA 032,165,204,164,165,204,032,032  <046>
450 DATA 032,165,207,208,167,167,032,164  <074>
460 DATA 186,013,165,207,208,167,032,165  <085>
470 DATA 032,032,032,167,032,032,032,165  <070>
480 DATA 163,163,167,167,032,165,013,165  <105>
490 DATA 204,186,167,032,163,163,163,165  <116>
500 DATA 207,032,032,032,165,207,163,163  <109>
510 DATA 167,032,165,013,205,164,164,206  <126>
520 DATA 032,032,032,032,204,165,032,032  <113>
530 DATA 032,205,163,163,208,167,032,165  <148>
540 DATA 013,032,032,032,032,032,032,032  <124>
550 DATA 032,032,032,032,032,032,032,163  <140>
560 DATA 163,163,032,163,013,017,017,018  <169>
570 DATA 032,032,032,032,032,032,032,032  <155>
580 DATA 032,032,032,032,032,032,032,032  <165>
590 DATA 032,032,032,032,032,032,032,032  <175>
600 DATA 032,074,079,089,083,084,073,067  <240>
610 DATA 075,032,080,065,073,078,084,069  <243>
620 DATA 082,032,032,032,032,032,032,032  <210>
630 DATA 032,032,032,032,032,032,032,032  <215>
640 DATA 032,032,032,032,032,032,032,032  <225>
650 DATA 032,032,013,017,017,144,032,040  <243>
660 DATA 067,041,032,049,057,056,052,032  <020>
670 DATA 032,066,089,032,067,046,083,065  <043>
680 DATA 085,069,082,013,017,017,029,029  <049>
690 DATA 029,029,029,018,156,032,032,032  <048>
700 DATA 084,065,083,084,069,032,032,032  <065>
710 DATA 013,000,169,013,141,000,144,169  <054>
720 DATA 044,141,001,144,169,021,141,002  <060>
730 DATA 144,169,019,141,003,144,169,205  <095>
740 DATA 141,005,144,162,189,138,157,000  <101>
750 DATA 016,169,006,157,000,148,202,224  <106>
760 DATA 255,208,242,096,169,020,133,177  <134>
770 DATA 169,000,133,176,160,000,145,176  <128>
780 DATA 230,176,208,002,230,177,166,177  <149>
790 DATA 224,032,208,242,096,134,176,132  <154>
800 DATA 177,138,041,248,133,178,165,176  <153>
810 DATA 056,229,178,133,180,165,177,041  <188>
820 DATA 240,133,179,165,177,056,229,179  <208>
830 DATA 133,181,169,000,133,176,133,182  <192>
840 DATA 169,020,133,177,162,021,165,179  <211>
850 DATA 032,083,034,202,208,248,165,178  <219>
860 DATA 024,010,168,169,000,101,182,133  <208>
870 DATA 182,152,032,083,034,169,000,133  <225>
880 DATA 182,165,181,032,083,034,056,169  <253>
890 DATA 007,229,180,170,240,009,169,001  <249>
900 DATA 010,202,208,252,133,182,096,169  <007>
910 DATA 001,208,249,024,101,176,133,176  <013>
920 DATA 165,182,101,177,133,177,096,032  <036>
930 DATA 245,033,160,000,177,176,005,182  <033>
940 DATA 145,176,096,032,245,033,160,000  <042>
950 DATA 165,182,073,255,049,176,145,176  <080>
960 DATA 096,032,245,033,160,000,177,176  <067>
970 DATA 037,182,197,182,208,002,160,001  <073>
980 DATA 096,160,048,032,071,032,165,250  <085>
990 DATA 208,008,136,208,246,160,000,132  <087>
```

Listing 3. Basic-Lader »Joypaint« (Teil 1)

```
1000 DATA 069,096,169,240,141,012,144,160  <111>
1010 DATA 037,162,255,234,234,234,234,234  <120>
1020 DATA 202,208,248,136,208,243,160,255  <130>
1030 DATA 032,071,032,165,250,208,013,136  <126>
1040 DATA 208,246,160,001,132,069,169,000  <142>
1050 DATA 141,012,144,096,169,144,141,012  <152>
1060 DATA 144,160,032,162,255,234,234,202  <159>
1070 DATA 208,251,136,208,246,160,002,208  <174>
1080 DATA 227,169,015,141,014,144,032,137  <184>
1090 DATA 034,192,000,240,006,165,069,201  <186>
1100 DATA 002,240,001,096,169,000,141,014  <186>
1110 DATA 144,032,137,034,192,002,240,249  <212>
1120 DATA 160,002,132,069,208,237,165,197  <236>
1130 DATA 201,039,240,009,201,047,240,014  <220>
1140 DATA 160,000,132,069,096,169,240,032  <245>
1150 DATA 032,035,160,001,208,007,169,144  <248>
1160 DATA 032,032,035,160,002,132,069,096  <002>
1170 DATA 141,012,144,169,015,141,014,144  <009>
1180 DATA 160,037,162,255,234,234,202,208  <029>
1190 DATA 251,136,208,246,169,000,141,012  <034>
```

Listing 3. Basic-Lader »Joypaint« (Teil 1, Schluß)

```
100 REM *** JOYPAINT TEIL 2                   <063>
110 REM >> VOR DEM LADEN <<                   <034>
120 REM >> POKE 44,39 EINGEBEN <<             <103>
130 FOR T=9016 TO 9825:READ D:S=S+D:POKE T,D
   :NEXT                                      <043>
140 IF S<>93663 THEN PRINT"{CLR,3DOWN}FEHLER
    IN DATAS !!":END                          <217>
150 SYS 9682                                  <013>
160 DATA 144,141,014,144,096,032,178,033  <030>
170 DATA 169,000,170,149,064,232,224,006  <038>
180 DATA 208,249,170,168,032,095,034,032  <058>
190 DATA 217,034,208,003,032,254,034,208  <052>
200 DATA 124,032,016,036,032,071,032,208  <055>
210 DATA 009,032,002,032,208,004,169,000  <060>
220 DATA 240,229,133,183,165,068,240,007  <096>
230 DATA 032,098,034,169,000,240,003,032  <088>
240 DATA 110,034,166,183,189,249,035,168  <129>
250 DATA 189,239,035,024,101,064,201,255  <122>
260 DATA 240,006,201,166,176,006,208,006  <124>
270 DATA 169,000,240,002,169,166,133,064  <140>
280 DATA 133,065,170,024,152,101,066,201  <137>
290 DATA 255,240,006,201,143,176,006,208  <155>
300 DATA 006,169,000,240,002,169,143,133  <161>
310 DATA 066,133,067,168,032,121,034,133  <180>
320 DATA 068,168,240,007,032,110,034,169  <192>
330 DATA 000,240,003,032,098,034,164,050  <184>
340 DATA 032,042,035,165,070,240,128,169  <211>
350 DATA 000,133,070,240,132,165,069,201  <206>
360 DATA 002,240,014,169,008,162,001,134  <215>
370 DATA 068,133,183,169,001,133,070,208  <244>
380 DATA 131,162,000,169,004,208,240,234  <239>
390 DATA 000,000,000,255,255,255,000,001  <227>
400 DATA 001,001,255,001,000,000,255,001  <228>
410 DATA 000,000,255,001,000,000,000,000  <223>
420 DATA 000,000,000,000,000,000,000,000  <220>
430 DATA 173,141,002,201,001,240,006,165  <021>
440 DATA 053,048,002,133,068,173,141,002  <046>
450 DATA 201,004,240,018,165,197,201,024  <054>
460 DATA 208,003,076,211,036,201,055,240  <062>
470 DATA 067,201,063,240,068,096,173,015  <091>
480 DATA 144,133,055,169,008,141,015,144  <095>
490 DATA 165,197,162,021,221,079,038,240  <113>
500 DATA 016,202,208,248,173,141,002,201  <101>
510 DATA 004,240,237,165,055,141,015,144  <119>
520 DATA 096,138,010,170,234,189,084,038  <151>
530 DATA 133,048,189,085,038,133,049,165  <168>
540 DATA 055,141,015,144,160,144,032,042  <143>
550 DATA 035,108,048,000,169,003,133,050  <155>
560 DATA 096,169,043,208,249,169,004,141  <194>
570 DATA 014,144,169,160,141,011,144,173  <180>
580 DATA 015,144,133,051,141,015,144,165  <186>
590 DATA 197,201,064,240,250,201,005,240  <192>
600 DATA 010,201,061,240,010,201,015,240  <178>
610 DATA 032,208,236,169,001,208,002,169  <226>
620 DATA 255,133,052,165,051,024,101,052  <224>
```

Listing 4. Basic-Lader »Joypaint« (Teil 2)

```
630 DATA 133,051,160,048,162,255,234,234  <246>
640 DATA 234,202,208,250,136,208,245,240  <252>
650 DATA 203,169,000,141,011,144,141,014  <244>
660 DATA 144,096,169,001,133,053,096,169  <036>
670 DATA 000,240,249,169,255,208,245,165  <041>
680 DATA 068,240,007,032,098,034,169,000  <041>
690 DATA 240,003,032,110,034,032,024,229  <026>
700 DATA 096,032,178,033,076,195,035,169  <083>
710 DATA 000,133,198,032,215,036,162,000  <055>
720 DATA 189,255,037,032,210,255,232,224  <084>
730 DATA 017,208,245,162,000,032,015,225  <077>
740 DATA 201,013,240,008,157,000,002,232  <072>
750 DATA 224,016,144,241,134,250,162,000  <094>
760 DATA 189,016,038,032,210,255,232,224  <120>
770 DATA 049,208,245,032,228,255,240,251  <137>
780 DATA 201,049,240,006,201,050,240,006  <118>
790 DATA 208,241,162,001,208,002,162,008  <136>
800 DATA 134,251,169,000,133,193,133,172  <127>
810 DATA 133,174,169,020,133,194,133,173  <176>
820 DATA 169,032,133,175,165,250,133,183  <189>
830 DATA 169,000,133,187,169,002,133,188  <201>
840 DATA 165,251,133,186,032,249,253,169  <218>
850 DATA 060,133,178,169,003,133,179,096  <227>
860 DATA 032,239,036,032,130,246,032,134  <212>
870 DATA 037,076,233,036,032,239,036,166  <241>
880 DATA 193,164,194,169,000,032,066,245  <252>
890 DATA 032,134,037,076,233,036,169,013  <253>
900 DATA 032,210,255,165,251,201,001,240  <240>
910 DATA 032,169,008,032,180,255,169,111  <018>
920 DATA 032,150,255,032,165,255,201,013  <013>
930 DATA 240,006,032,066,231,184,080,243  <029>
940 DATA 032,066,231,032,171,255,024,144  <038>
950 DATA 012,032,183,255,240,007,169,105  <051>
960 DATA 160,195,032,030,203,169,008,133  <060>
970 DATA 255,162,255,160,255,234,234,136  <084>
980 DATA 208,251,202,208,246,198,255,208  <098>
990 DATA 240,096,032,132,032,032,228,255  <090>
1000 DATA 201,000,240,249,169,027,141,015  <095>
1010 DATA 144,169,037,133,050,169,255,133  <126>
1020 DATA 053,165,254,201,210,240,003,032  <105>
1030 DATA 220,033,169,210,133,254,076,061  <132>
1040 DATA 035,234,234,234,234,234,234,234  <146>
1050 DATA 147,017,018,070,073,076,069,013  <163>
1060 DATA 018,078,065,077,069,058,146,032  <188>
1070 DATA 013,017,017,018,068,069,086,073  <187>
1080 DATA 067,069,058,146,032,049,046,032  <200>
1090 DATA 084,065,080,069,032,032,032,040  <190>
1100 DATA 049,041,013,032,032,032,032,032  <182>
1110 DATA 032,032,032,050,046,032,070,076  <200>
1120 DATA 079,080,080,089,032,040,056,041  <229>
1130 DATA 013,017,017,070,079,080,080,089  <241>
1140 DATA 032,069,082,082,079,082,058,032  <003>
1150 DATA 034,042,041,021,039,047,220,033  <239>
1160 DATA 125,036,104,037,116,037,202,036  <002>
1170 DATA 207,036                          <119>
```

Listing 4. Basic-Lader »Joypaint« (Teil 2, Schluß)

```
100 REM *********************************  <051>
110 REM ***                           ***  <249>
120 REM ***  MULTICOLOR-FIGUR  FUER   ***  <233>
130 REM ***  ALLE   AUSBAUVERSIONEN   ***  <165>
140 REM ***                           ***  <023>
150 REM *********************************  <101>
160 PRINT"{CLR}"                           <016>
170 FOR T=0 TO 7:READ D:POKE 7168+T,D:NEXT <018>
180 POKE PEEK(648)*256+69,0                <190>
190 REM *** @ IN BILDSCHIRMPOSITION        <248>
200 POKE 38469,10                          <250>
210 REM *** BEI 8 KBYTE ERWEITERUNG        <183>
220 REM *** DEN POKE DURCH 37957,10        <001>
230 POKE 36869,PEEK(36869)OR 15            <242>
240 REM *** ZEICHENGENERATOR INS RAM       <116>
250 GOTO 250                               <026>
260 DATA 48,68,204,48,100,48,48,136        <155>
```

Listing 5. Ein Multicolor-Demo

Stringprogrammierung in Maschinensprache Teil 2

Hier ist ein weiterer Beitrag über effektives Programmieren. Auch er beschäftigt sich mit Strings und enthält einige neue Tips zur Garbage Collection.

Diesmal geht es darum, wie man Stringfunktionen selbst programmieren kann. Hierbei werden wir streng darauf achten, die Müllstrings im Zaum zu halten, um unserem »Erzfeind«, der Garbage Collection, wenig Arbeit zu lassen.

Noch einmal: Garbage Collection

Ich möchte aber vorher noch etwas richtigstellen, was ich in der letzten Ausgabe wohl nicht klar genug ausgedrückt habe.

Von vielen Leuten hört man den Vorschlag, ab und zu per FRE(0) eine Garbage Collection auszulösen, um ein Ansammeln von Müllstrings und eine lange Garbage Collection zu vermeiden.

Wer allerdings den Beitrag aus der letzten Ausgabe aufmerksam gelesen hat, wird mir zustimmen: Dies ist absolut falsch! Denn die Dauer der Garbage Collection richtet sich ja hauptsächlich nach der Anzahl der definierten Strings und nicht nach der der Müllstrings. Die von Hand ausgelöste Garbage Collection ist also nur unwesentlich kürzer als die automatisch durchgeführte. Jede von Hand ausgelöste Müllabfuhr ist damit unnötig und kostet nur Zeit! Also: Lieber sehr viel Müll ansammeln lassen und dafür mit möglichst wenig Stringvariablen arbeiten. Eine Ausnahme ist natürlich klar. Sollten Sie zeitkritische Teile in einem Programm haben, in denen Sie sich gar keine Garbage Collection erlauben können, so lohnt es sich, kurz vorher PRINT FRE(0) einzugeben.

Problem: Strings auffüllen

Vielleicht haben auch Sie in einigen Programmen diese oder eine ähnliche Zeile entdeckt:

170 IF LEN(A$)<40 THEN A$=A$+" ": GOTO 170

Die Bedeutung ist klar: Hier soll der String A$ auf 40 Zeichen Länge aufgefüllt werden. Aber schon jetzt müßte es Ihnen eiskalt den Rücken herunterlaufen. Angenommen, A$ hätte zu Anfang 20 Zeichen, dann werden mindestens 20 Müllstrings durch die ständige Zuweisung erzeugt. Sollte diese Auffüllung öfter durchgeführt werden, darf man sich nicht wundern, wenn bald der ganze Speicher voll ist.

Eine weitere Lösung sieht meist so aus:

10 H$="40 * space"
170A$=A$+LEFT$(H$,40–LEN(A$))

Schon sehr viel besser! Es entstehen zwei Müllstrings (der alte Wert von A$ und einer, der durch die LEFT$-Funktion entsteht und nur Leerzeichen enthält). Es wird außerdem eine Variable H$ benötigt.

Kurz ein paar Worte zur Entstehung des zweiten Müllstrings. Stringfunktionen werden über einen String-Stack abgewickelt. Auf diesem werden die Stringdescriptoren der Zwischenergebnisse bei längeren Stringoperationen abgelegt. In unserem Fall wird zuerst der A$-Descriptor auf diesen String-Stack gelegt, dann der LEFT$-String erstellt und dessen Descriptor abgelegt. Danach erst wird die »+«-Verknüpfung durchgeführt, die einen neuen String erstellt. Die beiden Descriptoren werden vom String-Stack entfernt, und die zugehörigen Strings finden sich als Müll im Speicher wieder. Das Prinzip des String-Stacks ermöglicht eine hierarchische Abarbeitung von Stringfunktionen. Es gilt dabei die Regel:

Erst LEFT$, RIGHT$, MID$, dann + ähnlich dem »Punkt vor Strich« aus der Mathematik.

Deswegen dürfen auch bei Stringoperationen Klammern gesetzt werden.

Wir werden uns im folgenden nicht mehr mit dem String-Stack beschäftigen, weil Sie ihn bei selbstprogrammierten Funktionen wohl nie benötigen werden. Außerdem ist der Umgang mit ihm nicht ganz so einfach, wie es im ersten Augenblick klingt.

Nun aber zu unserem Beispiel. Will man auf der Basic-Ebene ohne PEEK und POKE arbeiten, ist die zweite Lösung die effektivste. Aber ich gebe mich, unersättlich wie ich bin, immer noch nicht zufrieden, denn es geht

1. noch etwas schneller und
2. mit nur einem einzigen Müllstring.

Dazu müssen wir aber auf die Maschinensprachen-Ebene herunter. Werfen Sie mal einen Blick auf Listing 1.

Diese FORMAT-Routine simuliert einen neuen Basic-Befehl. FORMAT füllt einen String mit Leerzeichen, bis eine definierte Länge erreicht ist. Wenn ein String länger ist, wird er abgeschnitten. Der Aufruf muß allerdings über einen SYS-Befehl erfolgen. Steht FORMAT beispielsweise im Kassettenpuffer, so führt

CLR: SYS 826 (A$,250,A$)

dazu, daß A$ 250 Leerzeichen enthält.

A$="HALLO": SYS 826 (A$,10,B$)

läßt A$ wie es war, B$ enthält aber »HALLO« und 5 angehängte Leerzeichen, hat also die Länge 10.

A$="TEST":N=1:SYS 826 (A$,N,B$(0))

hinterläßt in B$(0) ein einsames T, der Rest wird abgeschnitten.

Zusammengefaßt läßt sich also sagen, daß bei der Parameterübergabe:

(String1, N, String2)

in String2 genau N Zeichen aus String1 stehen, und daß gegebenenfalls String2 mit Leerzeichen aufgefüllt wird.

Nur wenn »String1« den gleichen Namen hat wie »String2«, entsteht ein Müllstring, nämlich der alte Inhalt der Stringvariablen. Für die, die es nun gar nicht mehr erwarten können, dieses Programm auszuprobieren, gibt es in Listing 2 einen Basic-Lader. Listing 1 kann direkt mit einem Assembler oder auch mit dem SMON eingegeben werden. Anhand dieses Programms, das alle wichtigen ROM-Routinen, die mit Strings zu tun haben, aufruft, wollen wir nun die Programmierung solcher Routinen erarbeiten.

FORMAT analysiert

Nehmen wir uns erst einmal die grundsätzliche Funktionsweise von FORMAT vor. Ein einfaches Flußdiagramm ist in Bild 1 dargestellt. Dies zeigt aber nur die Verfahrensweise von FORMAT. Die eingebauten Sicherheitsüberprüfungen sind hier nicht enthalten. Bekannt sind am Start der String1, seine Länge 1

```
JSR $AEFA      Klammer auf?
JSR $AD9E      Auswerten eines beliebigen Ausdrucks
JSR $B6A3      Weitere Auswertung für Strings
STX $FB        Stringadresse LO-Byte
STY $FC        Stringadresse HI-Byte
STA $FD        Länge des Strings
JSR $AEFD      Komma?
JSR $B79E      Hole Bytewert in X
TXA
JSR $B47D      Reserviere Speicherplatz für Endstring Adresse
               in $62/$63, Länge in $61
LDY #$00
LBL1
CPY $61        Länge des Endstrings erreicht?
BEQ LBL3
CPY $FD        Startstring komplett kopiert?
BEQ LBL2
LDA($FB),Y     kopieren des Startstrings
STA($62),Y     in den Endstring
INY
BNE LBL1       unbedingter Sprung
LBL2
LDA #$20       Leerzeichen zum Auffüllen
STA($62),Y     Auffüllen
CPY $61        Endstring voll?
BEQ LBL3
INY
BNE LBL2       Unbedingter Sprung
LBL3
JSR $AEFD      Komma?
JSR $B08B      Variable suchen/einrichten
LDX $0D
BEQ LBL5       Wenn kein String dann TYPE MISMATCH
STA $FB        Variablenadresse LO
STY $FC        Variablenadresse HI
LDX #$02
LDY #$02       drei Werte sind zu übertragen
LBL4
LDA $61,X      kopieren des Descriptors von Endstring
STA($FB),Y     in die Stringvariable
DEX
DEY
BPL LBL4
JMP $AEF7      Klammer zu? Rücksprung zu Basic
LBL5
JMP $AD99      »TYPE MISMATCH ERROR«
```

Listing 1. Die FORMAT-Routine (nähere Erklärung im Text).

```
10 REM    *** FORMAT-ROUTINE ***           <177>
20 REM BRINGT STRINGS AUF DEFINIERTE       <078>
30 REM LAENGEN, FUELLT GGF. AUF.           <171>
40 REM                                     <183>
50 REM                                     <193>
60 REM SYNTAX:                             <236>
70 REM SYS ADRESSE(STARTSTR,N,ENDSTR)      <041>
80 :                                       <138>
90 :                                       <148>
100 REM DIESE ROUTINE IST FREI IM          <047>
110 REM SPEICHER VERSCHIEBLICH !!!         <110>
120 :                                      <178>
130 DATA 032,250,174,032,158,173,032,163   <254>
140 DATA 182,134,251,132,252,133,253,032   <003>
150 DATA 253,174,032,158,183,138,032,125   <027>
160 DATA 180,160,000,196,097,240,022,196   <033>
170 DATA 253,240,007,177,251,145,098,200   <044>
180 DATA 208,241,169,032,145,098,196,097   <076>
190 DATA 240,003,200,208,247,032,253,174   <050>
200 DATA 032,139,176,166,013,240,019,133   <071>
210 DATA 251,132,252,162,002,160,002,181   <062>
220 DATA 097,145,251,202,136,016,248,076   <101>
230 DATA 247,174,076,153,173,000,000,000   <087>
240 :                                      <042>
250 :                                      <052>
260 INPUT "STARTADRESSE";SA                <050>
270 FOR I=SA TO SA+85                      <110>
280 READ A:POKE I,A                        <104>
290 NEXT I                                 <238>
300 END                                    <173>
```

Listing 2. Basic-Lader der FORMAT-Routine

und die gewünschte Länge des String2.

Im ersten Schritt werden N Bytes für den String2 reserviert. Sodann werden solange Zeichen vom String1 in den String2 kopiert, bis entweder der String1 komplett kopiert wurde, oder der String2 schon voll ist. Im ersten Fall wird dann in einer zweiten Schleife der String2 mit Leerzeichen aufgefüllt. Ganz zum Schluß wird der Descriptor der zweiten Stringvariablen auf den String2 gerichtet. Das klingt alles ganz einfach, die Realisierung nach diesem Schema ist jedoch etwas umfangreicher.

Parameterübergaben

Sehen wir uns nun die ersten Zeilen des Listing 1 an. Der erste Befehl ist ein Sprung nach $AEFA. Dort steht eine ROM-Routine, die überprüft, ob als nächstes Zeichen ein »(« folgt. Dies ist an sich nicht notwendig, trägt aber erheblich zur Übersichtlichkeit solcher Routinen bei. Fehlt das »(«, so wird SYNTAX ERROR ausgegeben.

Die nächsten zwei Sprungbefehle gehören zusammen. $AD9E wertet einen beliebigen Term, Zahlenrechnung oder String aus und hinterläßt wichtige Parameter für $B79E. Diese Routine prüft, ob der vorherige Term ein String war, und stellt dann im X-Register die LO- und im Y-Register die HI-Adresse des resultierenden Strings, sowie im Akku die Länge des Strings bereit. Diese beiden Sprungbefehle werten auch Ausdrücke wie LEFT$(A$+B$,8) oder ähnliche aus, so daß auch solche Ausdrücke an selbstentwickelte Routinen weitergegeben werden können. Auch können Array-Werte wie A$(14) übergeben werden, ohne daß eine Spezialbehandlung nötig wäre. TYPE MISMATCH und ähnliche Fehlermeldungen werden vollautomatisch ausgegeben.

Die soeben gewonnenen Parameter des ersten Strings werden nun zwischengespeichert. $AEFD prüft auf ein Komma, und verhält sich ansonsten genauso wie $AEFA.

Mit $B79E wird ein Ein-Byte-Wert, das heißt eine Zahl zwischen 0 und 255, in das X-Register geholt. Bei größeren Zahlen wird ILLEGAL QUANTITY angezeigt. Auch hier dürfen wieder Berechnungen oder Variablen stehen.

Damit wäre die Parameterübergabe vorläufig beendet, die weiteren Parameter für String2 besorgen wir uns erst, wenn wir sie tatsächlich brauchen.

Erstellen von String2

Erinnern wir uns noch einmal an den Artikel aus der letzten Ausgabe. Dort wurde gesagt, daß Strings im Speicher von oben nach unten wachsen, während es beim Programm und bei den Variablen genau umgekehrt ist.

An sich müßten wir nun an der unteren Grenze der Strings genügend Bytes für String2 durch Verändern der entsprechenden Pointer herstellen. Es geht aber auch einfacher. $B47D reserviert so viele Bytes an der entsprechenden Stelle, wie ihr im Akku übergeben werden. Es wird sogar, wenn nötig, eine Garbage Collection durchgeführt, und schlimmstenfalls OUT OF MEMORY angezeigt, falls der Speicherplatz nicht reicht. Nach dem Aufruf der Routine $B47D stehen in den Bytes $62/$63 die Startadressen für den neuen String und in $61 dessen Länge, die wir ja festgelegt haben. Wenn Sie genauer hinschauen, bemerken Sie, daß

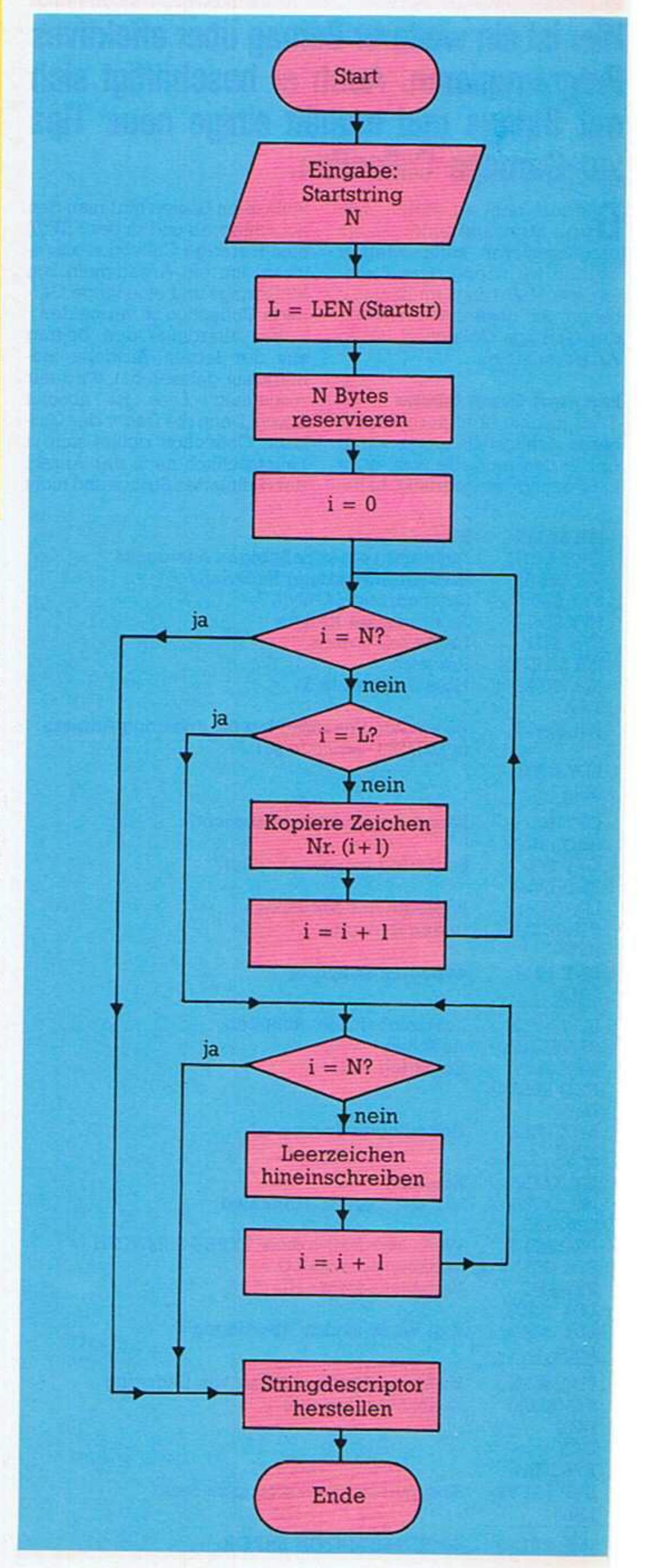


Bild 1. Flußdiagramm der FORMAT-Routine

dies schon der komplette Descriptor ist, der später nur noch in die Stringvariable, die String2 enthalten soll, einkopiert werden muß. Somit ist es recht einfach, String2 zu erstellen, da wir ja jetzt seinen Bestimmungsort kennen und die Länge für spätere Vergleiche zwischengespeichert wurde. Diese Routine ist wohl der Kernpunkt eines jeden Maschinenprogramms, das Strings verwalten soll.

Nun folgen zwei Schleifen, die genauso aufgebaut sind, wie die entsprechenden Teile im Flußdiagramm. Die erste kopiert Zeichen von String1 in String2, die zweite füllt String2 mit Leerzeichen auf.

Sollten Sie die etwas unlogische Struktur oder die beiden unbedingten Sprünge BNE nach den Inkrementanweisungen stören: Dieses Programm durfte, damit es frei verschiebbar ist, keine JMP-Befehle enthalten. Außerdem werden ohne Spezialabfrage sämtliche Sonderfälle (Länge eines oder beider Strings gleich Null) berücksichtigt.

Parameterübergabe II

Nachdem String2 fertiggestellt wurde, folgt der abschließende Teil, der den Descriptor von String2 in die entsprechende Stringvariable hineinkopieren soll.

Doch zuerst wieder der Test auf ein Komma. Die Routine $B08B nimmt die Schlüsselposition in der Übergabe von Strings an den Basic-Text ein. Sie versucht eine Variable im Speicher und gibt deren Adresse im Akku (LO-Byte) und Y-Register (HI-Byte) zurück. Leider funktioniert diese Routine auch mit numerischen Variablen. Die nächsten beiden Befehle fangen diesen Fall ab. In der Zelle $0D steht (handelt es sich um eine Zahl), $00, bei einem String $FF. Haben wir keine Stringvariable vor uns, so müssen wir selber in die Routine springen, die TYPE MISMATCH ERROR ausgibt ($AD 99). Falls eine gesuchte Variable noch nicht existiert, wird sie in der Routine $B08B angelegt.

Anschließend speichern wir in den zwei nun überflüssig gewordenen Speicherstellen $FB/$FC die Adresse der Variablen ab. Dies ist an sich nicht die genaue Adresse der Variablen, sondern, zu unserem Glück, die Adresse, an der der Stringdescriptor stehen muß. Im vorletzten Schritt kopieren wir also den schon lange vorhandenen Descriptor in die entsprechenden Speicherstellen. Somit ist die Arbeit faktisch beendet, und wir prüfen noch, weil's ordentlicher aussieht, ob auch ein »)« vorhanden ist. Dadurch, daß wir mit JMP springen, ersparen wir uns auch noch ein RTS, das ja am Ende der Klammerzu-Betriebssystem-Routine steht.

Ich glaube, daß ich mit diesem Beispiel das angenehme mit dem nützlichen verbunden habe, und hoffe, daß Sie jetzt sowohl ein wenig schlauer, als auch um ein nützliches Programm reicher sind.

So nicht !

Bevor noch einmal alle wichtigen ROM-Routinen zusammengefaßt werden, möchte ich Ihnen ein Beispiel geben, wie man sehr leicht Fehler in eine Stringfunktion einbauen kann, die nicht sofort erkennbar sind. Auch hier ist ein Blick auf die Garbage Collection angebracht.

Wie oben schon erklärt wurde, erzeugt die Zuweisung
B$=LEFT$(A$,3)
einen zweiten String, der nur die drei ersten Zeichen von A$ enthält und auf den Descriptor von B$ zeigt. Manche Leser mögen jetzt folgende Ideen haben:

Theoretisch ist es ja nicht notwendig, einen zweiten String mit den drei Zeichen zu erstellen. Vielmehr könnte man ja den Descriptor von B$ auf dieselbe Position wie den von A$ zeigen lassen. Im B$-Descriptor wird dann die Stringlänge 3 angegeben, während die in A$ gleich bleibt.

Sodann wäre der Teilstring B$ in A$ enthalten und müßte nicht noch einmal im Speicher stehen. Ähnliches ließe sich auch mit den Routinen erreichen, die dann RIGHT$ und MID$ ersetzen.

Doch vielleicht ahnt der eine oder andere unter Ihnen schon den Haken an dieser Geschichte. Es ist wieder einmal (was sonst) die Garbage Collection, die uns einen Strich durch die Rechnung macht.

Findet, aus welchem Grund auch immer, eine Garbage Collection statt, so passiert folgendes: A$ und B$ zeigen auf denselben Stringdescriptor. Je nachdem welcher String als erster im Programm definiert wurde, wird einer der beiden zuerst aufgeräumt, nehmen wir mal an, es sei B$. Wenn B$ nun aufgeräumt ist, wird sein Inhalt an die aktuelle »Aufräumgrenze« hochkopiert. Diese aktuelle Aufräumgrenze kann allerdings so ungünstig liegen, daß A$ von B$ teilweise überschrieben wird. Im nächsten Aufräumschritt würde A$ nicht an die aktuelle Aufräumgrenze hoch, sondern unter sie herunter kopiert und somit weitere Strings zerstört... Dieses Spielchen würde sich dann bis zum Ende der Garbage Collection hinziehen. (Oder bis einmal tatsächlich Müllstrings an den kritischen Stellen stehen, die ja überschrieben werden dürfen).

Zugegeben, ich habe mir gerade den Worst Case, den allerschlimmsten Fall, herausgesucht. Er zeigt aber ganz deutlich folgende Grundregel, die beim Programmieren von Stringfunktionen beachtet werden soll:

Zwei Stringdescriptoren dürfen niemals auf denselben String zeigen, sonst kann die Garbage Collection Stringinhalte zerstören.

Sie können sich ja spaßeshalber vor Augen führen, was man so alles an Stringsalat produzieren kann, indem Sie mal solch eine Routine programmieren. Mit den oben gegebenen Informationen dürfte das nicht schwerfallen.

Noch fehleranfälliger als die neue LEFT$-Routine wären solche, die RIGHT$ und MID$ ersetzen sollen, weil da nämlich sowohl Länge als auch Adresse im Stringedescriptor unterschiedlich sind. Dann kann eine Garbage Collection zum völligen Chaos führen, es entsteht im wahrsten Sinne des Wortes Stringmüll.

In solchen Fällen muß man, wie in unserer FORMAT-Routine, einfach den Teil des Strings kopieren, der weiterbearbeitet werden soll. Dies machen ja schließlich auch die originalen LEFT$-, RIGHT$- und MID$-Routinen.

Bild 2 enthält noch einmal eine Kurzbeschreibung aller verwendeten Routinen, damit Sie bei der Programmierung von Funktionen nicht immer in der Beschreibung des Beispiels nachsehen müssen.

Mit diesen Routinen läßt sich meiner Ansicht nach jede nur erdenkliche Stringmanipulation durchführen, auch wenn der Aufwand teilweise durch weitere ROM-Routinen eingeschränkt werden kann. Aber das Arbeiten über die Descriptoren der zu bearbeitenden Strings selbst, wie wir es in unseren Beispielen immer gemacht haben, reicht völlig aus. Auch benötigen wir nicht den schon erwähnten String-Stack. Wer trotzdem mehr über die interne Stringverarbeitung des C 64 wissen möchte, der sollte sich ein ROM-Listing zur Hand nehmen und versuchen, die einzelnen Routinen nachzuverfolgen.

So, ich glaube für heute reicht's wirklich. Aber ich entlasse Sie nicht, ohne Ihnen einen Vorgeschmack auf die nächste Ausgabe zu geben. Dort werden wir uns etwas genauer mit Arrays beschäftigen, allerdings nicht gar so ausführlich wie mit den Strings. Im Mittelpunkt steht nämlich das Suchprogramm INTELLISEARCH. Dieses Suchprogramm bietet, neben einem sehr schnellen Suchalgorithmus, viele Besonderheiten, die jede INSTR- oder ähnliche Funktion vor Neid erblassen läßt. Das Ganze natürlich ausführlich dokumentiert. (Boris Schneider/gk)

$AEFA	Prüft aktuelles Zeichen auf »Klammer auf«, sonst SYNTAX ERROR
$AEF7	Prüft auf »Klammer zu«
$AEFD	Prüft auf »Komma«
$AD9A	Vorauswertung eines beliebigen Ausdrucks (auch verschachtelt und verklammert), weitere Auswertung bei Strings immer mit $B6A3
$B6A3	Weitere Auswertung eines Stringausdrucks, nachher: Akku: Länge des Strings, X-Reg: Adresse LO-Byte, Y-Reg: Adresse HI-Byte
$B79E	Holt Byte-Wert, das heißt Zahl zwischen 0 und 255 ins X-Reg, kann auch Ergebnis einer Rechnung oder Funktion sein.
$B47D	Reserviert Speicher für einen String am Ende des Stringspeichers, Anzahl der Zeichen muß vorher im Akku stehen. Nachher stehen in $61: Länge des reservierten Bereichs, $62: Adresse LO-Byte, $63: Adresse HI-Byte des reservierten Bereichs.
$B08B	Holt sich Variablennamen aus Basic-Text, sucht diese Variable im Speicher. Ist sie nicht vorhanden, wird sie automatisch angelegt. Es stehen dann in: $0D: Das Typflag dieser Variablen. (Strings = $FF, Zahl = $00), Akku: LO-Byte, Y-Reg: HI-Byte der Adresse, an der, war es ein String, der Stringdescriptor beginnt.
$AD99	Gibt Fehlermeldung TYPE MISMATCH aus.

Bild 2. Alle im Listing 1 verwendeten Betriebssystem-Routinen

MEMORY MAP

Nur zwei Adressenpaare wollen wir diesmal behandeln. Die haben es aber in sich.

Das letzte Mal haben wir den Zeiger in den Speicherzellen 43-44 ($2B—$2C) kennengelernt, der den Anfang des Speicherbereichs für Basic-Programme angibt.

Die anschließenden Speicherzellen-Paare bis 55-56 ($37—$38) zeigen auf weitere für Basic-Programme wichtige Speicherbereiche, die deswegen gemeinsam betrachtet werden sollten. Bild 1 stellt den Zusammenhang grafisch dar. Dabei ist wichtig zu wissen, daß ein Basic-Programm während des Eintippens oder Einladens von Disk beziehungsweise Kassette in den ersten Block kommt. Während des Programmlaufs werden alle normalen Variablen in den Block geschrieben, alle Felder (Arrays) in den zweiten Block und schließlich der Text der Zeichenketten (Strings) sozusagen rückwärts vom Ende des Arbeitsspeichers (Block). Je nach Größe des Programms und nach Anzahl der Variablen wandern die Blockgrenzen nach oben beziehungsweise die von Block 4 nach unten. Wenn sie sich treffen beziehungsweise überschneiden gibt es »OUT OF MEMORY«.

Diese Blockbewegung ist in Bild 1 durch die Pfeile dargestellt.

Ich bin mir bewußt, daß gerade dieses Thema in den letzten Ausgaben des 64'er sowohl im Kurs »Der gläserne VC 20« als auch im Assembler-Kurs behandelt worden ist. Ich bin aber der Meinung, daß man es gar nicht oft genug erklären kann und nehme daher eine gewisse Wiederholung in Kauf.

Adresse 45-46 ($2D-$2E)
Zeiger auf die Anfangsadresse des Speicherbereichs für Variable

Dieser Zeiger, in der Low/High-Byte-Darstellung, gibt dem Basic-Interpreter an, ab welcher Speicherzelle die Variablen eines Basic-Programms gespeichert sind. Da die Variablen direkt an das Basic-Programm anschließen, zeigt dieser Zeiger natürlich gleichzeitig auf das Ende des Basic-Programms.

Es muß betont werden, daß es sich nur um den Bereich der »normalen« Variablen handelt, also nicht um Felder (Arrays). Anders als der Zeiger in 43-44, der auf fest definierte Speicherzellen zeigt, liegt der Zeiger für den Variablen-Beginn nicht fest. Je nach Länge des Programms wandert er nach oben.

Sobald ein Programm eingetippt oder aus einem externen Speicher (Diskette, Kassette) eingelesen ist, wird der Zeiger in 45-46 durch RUN auf ein Byte hinter das Programmende gesetzt und alle Variablen werden in der Reihenfolge ihres Auftretens gespeichert. Da normalerweise die Länge eines Basic-Programms während des Ablaufs konstant bleibt, werden die Variablen in ihrer Position auch nicht gestört.

Das bedeutet, daß sie sowohl vom Programm als auch vom Programmierer nach einer Unterbrechung abgefragt werden können. Nur wenn das Programm modifiziert wird, wandert der Zeiger zusammen mit den Variablen entsprechend weiter.

Den selben Effekt wie das oben erwähnte RUN haben übrigens auch die Befehle NEW, CLR und LOAD. Eine Ausnahme bildet das LOAD innerhalb eines Programms, welches den Zeiger nicht zurücksetzt. Dadurch wird ein Aneinanderhängen von mehreren Programmen samt Variablen-Weiterverwendung unter bestimmten Voraussetzungen ermöglicht.

Die Bearbeitung der Variablen durch das Basic-Programm und die daraus resultierenden Kochrezepte für den Programmierer sind in Tabelle 1 »Zwei Regeln für normale Variable« separat erläutert.

Die verschiedenen Typen der Variablen und ihre Darstellung im Speicher finden Sie im 64'er, Ausgabe 10/84, Seite 157 und noch ausführlicher in Ausgabe 11/84, Seite 124 dargestellt und erklärt.

Für diejenigen Leser, welche kein Monitor- beziehungsweise Disassembler-Programm haben oder benutzen können, ist in Tabelle 2 »Darstellung der normalen Variablen« eine kleine Anleitung gegeben, wie sie die Variablendarstellung mittels Basic anschauen können.

Adresse 47-48 ($2F-$30)
Zeiger auf die Anfangsadresse des Speicherbereichs für Feld (Arrays)

Die Darstellung und Behandlung der Felder werden in der nächsten Folge in gleicher Art und Weise behandelt, wie die normalen Variablen.

(Dr. H. Hauck/aa)

Zwei Regeln für »normale« Variable

Alle Daten, die in einem Basic-Programm nicht in Form von READ-DATA-Anweisungen vorkommen, werden als »Variable« unmittelbar nach dem Basic-Programm abgespeichert. Wir unterscheiden dabei zwei Typen:
— normale Variable
— Felder (Arrays)

Wir betrachten hier nur die »normalen« Variablen.

Sie erscheinen in dem Speicherbereich, dessen Beginn durch den Zeiger in den Zellen 45-46 und dessen Ende durch den Zeiger in 47-48 angegeben wird, in derselben Reihenfolge, in welcher sie während des Ablaufes des Basic-Programms auftreten. Wenn Basic dann auf eine der bereits definierten (und abgespeicherten) Variablen zurückgreifen soll, muß es den gesamten Variablenbereich von Anfang an absuchen, bis es den Namen der gesuchten Variablen gefunden hat. Wenn diese Variable ganz am Ende des Bereiches steht, kann dieser Suchprozeß recht lange dauern.

Regel 1:
Häufig vorkommende Variable sollen am Anfang des Variablenbereichs stehen. Das wird dadurch erreicht, daß sie als erste Variable in einem Programm »definiert« werden. Falls sie erst später im Programm gebraucht werden (aber dann häufig), werden sie trotzdem am Anfang des Programms angegeben, notfalls mit einem beliebigen Wert, der später dann keine Rolle mehr spielt und ersetzt wird. Man nennt das einen »Dummy«-Wert.

Die Felder-Variablen stehen direkt nach den »normalen« Variablen. Auch hier kann der gewiefte Programmierer Gutes tun. Wenn nämlich nach einer Definition eines Feldes später im Programm noch normale Variable dazukommen, ist natürlich zuerst kein Platz für sie da. Das Betriebssystem des Computers muß erst alle Felder-Variablen weiterschieben, bevor die Neuankömmlinge in dem dadurch erweiterten Variablenbereich gespeichert werden können. Auch das kostet unnötig viel Zeit.

Regel 2:
Alle normalen Variablen sollen als erste in einem Programm definiert werden. Wer also drauflos programmiert, sollte zumindest am Ende das Programm so umbauen, daß diese simple Regel erfüllt wird.

Darstellung der normalen Variablen

Die normalen Variablen kommen in drei Arten vor:
— ganzzahlige Variablen
— Gleitkomma-Variablen
— String-Variablen (Zeichenketten)

Der Unterschied zwischen den drei Typen ist in den Commodore-Handbüchern gut erklärt, und ich verzichte hier auf eine Wiederholung. Ich will vielmehr direkt zeigen, wie die Variablen im Speicher abgelegt sind.

Wir können den Speicher direkt sichtbar machen.

Einmal geht das in Maschinencode mittels eines Monitors beziehungsweise Disassemblers.

Zum anderen aber geht das auch in Basic und zwar mit folgendem Trick, den ich Th. und M.L. Beyer (MC 10/1983) abgeschaut habe. Er wird im folgenden allerdings nur für den C 64 beschrieben, die Methoden für den VC 20 zeige ich das nächste Mal. Der Inhalt der Darstellung ist jedoch bei beiden Rechnern identisch, nur die »Sichtbarmachung« ist verschieden.

Wir verlegen den Beginn des Basic-Variablenspeichers einfach auf den Beginn des Bildschirmspeichers. Auf diese Weise können wir zwar kein vernünftiges Programm laufen lassen, aber alle direkt eingegebenen Variablen-Definitionen werden sofort sichtbar, weil sie eben im Bildschirmspeicher stehen.

Wir erreichen die Verlegung des Speichers durch »Verbiegen« der Zeiger in den Zellen 45-46 und 47-48. Die Bedeutung dieser Zeiger ist ja im Artikel erklärt.

mit Wandervorschlägen Teil 4

Beim C 64 beginnt der Bildschirmspeicher ab 1024. In Low/High-Byte Darstellung ist das 0/4 (1024/256 = 4, Rest 0). Geben Sie bitte direkt ein: POKE 46,4 :POKE 48,4

Das Low-Byte in 45 und 47 können wir weglassen, da es ja in beiden Fällen 0 ist. Wenn Sie jetzt den Bildschirm löschen, den Cursor ungefähr in die Mitte des Bildschirms fahren und wiederum direkt eingeben: VARIABLE = 3 und die RETURN-Taste drücken, dann erscheinen oben sieben Zeichen. Bitte schalten Sie mit der SHIFT- und Commodore-Taste auf den zweiten Zeichensatz um, jetzt können wir besser lesen. Aus anderen Kursen wissen Sie wahrscheinlich, daß Variable mit sieben Byte dargestellt werden. In der Tat sehen wir oben die ersten beiden Buchstaben des Variablennamens VA und fünf weitere Zeichen. Wir wollen aber systematisch vorgehen und uns zuerst die ganzzahligen Variablen anschauen.

Ganzzahl-Variable

Wiederholen Sie bitte den Vorgang (Löschen, Cursor auf Mitte, 2. Zeichensatz). Jetzt geben Sie eine Ganzzeit-Variable ein: VA% = 3

Nach RETURN sehen wir als erstes Zeichen ein reverses V, dann ein reverses A, den Klammeraffen @, das kleine c und nochmals zwei @. Die beiden ersten Zeichen des Variablennamens (besteht er nur aus einem Zeichen, wird mit einer 0 aufgefüllt) werden mit ihrem ASCII-Code eingegeben, zu dem bei Ganzzahl-Variablen zur Kennzeichnung einer solchen die Zahl 128 addiert wird.

Schauen Sie in der ASCII-Tabelle (64'er, Ausgabe 7/84) nach: Das V hat 86, um 128 erhöht gibt das 214. Wir arbeiten hier aber im Bildschirmspeicher, der die Zahlen auf seine eigene Weise interpretiert, nämlich als Bildschirmcode. Der Bildschirmcode-Tabelle entnehmen wir das Zeichen für den Wert 214, und das ist das invertierte V. Für das A können Sie das selbst nachvollziehen. Also: In unserer Darstellung erkennen wir Ganzzahlvariable an den invertierten Zeichen des Namens. Das 3. und 4. Zeichen sind das High- und Low-Byte des Variablenwertes und zwar im Bildschirmcode. In unserem Beispiel der 3 ist das High-Byte 0, also der Klammeraffe @, das Low-Byte 3, also das c. Die restlichen drei Bytes sind mit 0 aufgefüllt. Wenn Sie mit dem Cursor auf die 3 fahren, es mit einer 5 überschreiben und RETURN drücken, verwandelt sich das c in ein e. Beim Überschreiben mit 255 erscheint als 4. Byte das Zeichen für den Bildschirmcode 255. Beim Überschreiben mit 257 ändern sich beide Bytes. Das 3. (High-)Byte springt auf a (= 1), das 4. (Low-)Byte ebenfalls auf a. Nun, 1x256 + 1 = 257. Während, wie bewiesen, das Low-Byte von 0 bis 255 gehen kann, sind beim High-Byte nur Werte zwischen 0 und 127 zugelassen. Die Werte ab 128 signalisieren negative Zahlen. Probieren Sie es aus: 127x256 + 255 = 32767

Ein Überschreiben mit 32767 resultiert in einer Darstellung der Zeichen für den Bildschirmcode 127 und 255. Der Wert 32768 wird nicht mehr akzeptiert.

Negative Zahlen

Überschreiben Sie bitte die letzte Zahl mit 0. Wie zu erwarten war, sind Byte 4 und 5 jetzt 0 (Klammeraffe). Wenn Sie jetzt mit —1 überschreiben erscheint für beide Bytes das Zeichen mit dem Bildschirmcode 255. Bei —2 sehen wir die Zeichen mit den Code-Werten 255 und 254. Sie sehen also, daß die negativen Zahlen für ganzzahlige Variable sozusagen vom Ende der Tabelle her dargestellt werden, wobei die höchste negative Zahl wieder 32767 ist. Diese »Rückwärtszählung« ist bedingt durch die Methode der negativen Zahlendarstellung im Zweierkomplement. Der Platz und die Gelegenheit verbieten es mir, näher darauf einzugehen. Aber ich glaube, unser kleines Experiment hat Ihnen zumindest von der Darstellung her den Zusammenhang gezeigt. Im folgenden die Zusammenfassung der ganzzahligen Variablen.

1	2	3	4	5	6	7
Erstes	Zweites	High-	Low-			
Zeichen des Variablen-Namens (ASCII-Wert + 128)		Byte des Variablen-wertes		0	0	0

Gleitkomma-Variable

Ich hoffe, Sie verzeihen mir, wenn ich diese Darstellung heute überspringe. Sie ist nämlich nicht ganz leicht zu verstehen, und ich möchte sie lieber dann im Detail erklären, wenn wir zur Diskussion der Speicherzellen 97-101, nämlich des Gleitkomma-Akkumulators kommen. Da geht es in einem Stück. Als Vorgeschmack gebe ich jetzt nur die Zusammenfassung an.

1	2	3	4	5	6	7
Erstes	Zweites					
Zeichen des Variablen-Namens (ASCII-Wert)		Exponent + 129	Mantisse mit Genauigkeit von 32 Dualstellen, 1. Bit des 1. Bytes ist das Vorzeichen			

String-Variable

Zuerst ist es erforderlich, den Computer in den Anfangszustand zurückzusetzen. Wenn Sie einen RESET-Schalter haben, bitte diesen drücken, sonst aber aus- und einschalten. Wir geben nach Löschen des Bildschirms in der unteren Hälfte direkt ein: ZX$ = "A" <RETURN>

Fahren Sie bitte jetzt mit dem Cursor auf das A und ändern den String um in BC. Nach RETURN verwandelt sich das a in das b, das 4. Zeichen ebenfalls. Die ersten beiden Zeichen sind also wieder der Name der Variable. Um zu kennzeichnen, daß es eine String-Variable ist, erscheint das 2. Zeichen des Namens invertiert. Wie oben entsteht es dadurch, daß zum ASCII-Code die Zahl 128 addiert wird. Diese Zahl wird aber wieder als Bildschirmcode interpretiert und entsprechend angezeigt (vergleichen Sie es mit den ASCII- und Bildschirmcode-Tabellen). Das dritte Zeichen gibt die Länge des Strings an, also im ersten Fall mit a (= 1 im Bildschirmcode), im 2. Fall mit b (= 2). Zeichen 4 und 5 geben als Low- und High-Byte die Adresse an, bei der begonnen wird, den Text des Strings zu speichern. Wir hatten die beiden Fälle:

1. ZX $ = "A"

Viertes Zeichen: (Bildschirmcode: 255) und 5. Zeichen: (Bildschirmcode 156) ergibt als Adresse 40959.

2. ZX$ = "BC"

Viertes Zeichen: (Bildschirmcode 253) und 5. Zeichen: (Bildschirmcode 156) ergibt als Adresse 40957.

Der Text der Zeichenketten wird am Ende des Arbeitsspeichers (40959 beim C 64) abgelegt und zwar von hinten nach vorn. Mit PRINT PEEK(40957);PEEK(40958);PEEK(40959) drucken wir den Inhalt dieser Speicherzellen aus und erhalten: 66 67 65. Im ASCII-Code ist das: B C A. Die Zusammenfassung für String-Variable sieht so aus:

1	2	3	4	5	6	7
Erstes	Zweites		Low-	High-		
Zeichen des Variablen-Namens		Anzahl der Zeichen des Strings	Byte der Adresse, ab welcher der Text des Strings abgespeichert ist		0	0
ASCII-Wert	ASCII-Wert + 128					

Dem Klang auf der Spur Teil 3

Der SID (Sound Interface Device) ist funktionell an das klassische Konzept von Moog angelehnt. Dadurch ist seine Arbeitsweise leicht verständlich. Bild 1 zeigt das Blockschema des SID. Dieses Schema dient dazu, den Fluß der Audio-Signale zu veranschaulichen. Die Funktionsblöcke waren in ähnlicher Form bereits Bestandteile des Synthesizer-Schemas aus Teil 2. Den linken Teil des Schemas bilden drei identische, voneinander unabhängige, Funktionsgruppen.

Eine solche Gruppe setzt sich aus einem DCO (Digital Controlled Oscillator), einem Amplitudenmodulator (oder auch DCA = Digital Controlled Amplifier) und einem Hüllkurvengenerator (EG = Envelope Generator) zusammen. Der DCO kann wahlweise eine von vier Kurvenformen erzeugen: Dreieck, Sägezahn, Rechteck und Rauschen. Dabei ist das Tastverhältnis der Rechteckkurve steuerbar. Unter dem Tastverhältnis versteht man das Verhältnis zwischen der Länge des Kurvenabschnitts mit hoher Spannung zur gesamten Periodenlänge. Eine symmetrische Rechteckkurve hat demnach ein Tastverhältnis von 50 Prozent. Eine Veränderung im Tastverhältnis T bewirkt eine Klangfarbenänderung. So klingt eine Rechteckkurve mit T = 50 Prozent voluminös. Bewegt man sich mit T in Richtung 0 Prozent oder 100 Prozent, so wird der Klang zunehmend obertonreicher aber dünner, da der Anteil des Grundtons abnimmt. Einen besonders lebendigen Klang erhält man durch dynamische Veränderung des Tastverhältnisses, was zwar in der SID-Hardware nicht realisiert, aber softwaremäßig möglich ist.

Funktioneller Aufbau des SID

Der Hüllkurvengenerator beeinflußt über den Amplitudenmodulator den zeitlichen Lautstärkeverlauf der vom DCO kommenden Kurve. Die Hüllkurve wird nach dem bekannten ADSR-Schema parametrisiert.

Synchronisation und Ringmodulation

Die senkrechten Verbindungen von DCO1 zu DCO2, von DCO2 zu DCO3 und von DCO3 zurück zu DCO1 können einzeln zu- oder abgeschaltet werden. Sie dienen Spezialeffekten, die das Spektrum des SID beträchtlich erweitern. Im Normalfall, wenn diese Steuerpfade unwirksam geschaltet sind, schwingen die drei DCOs unabhängig voneinander, jeder in seiner vorprogrammierten Frequenz und Kurvenform. Im Falle der **Synchronisation** zwingt der synchronisierende DCO einen weiteren DCO dazu, gleichphasig zu schwingen. In Bild 2 erzeugt DCO1 eine Sägezahnschwingung von 100 Hz und synchronisiert DCO2, der ebenfalls auf Sägezahn, aber 350 Hz eingestellt ist. Nach jeweils dreieinhalb Perioden wird DCO 2 gezwungen, eine neue Periode anzufangen. DCO2 erzeugt so eine viel komplexere Kurvenform, als er es ohne Synchronisation tun würde. Variiert man nun noch die Frequenz eines der beiden DCOs, während man die des anderen konstant hält, so erzeugt DCO2 ständig andere Kurvenformen. Man erhält damit schwer zu beschreibende, aber meistens »elektronisch« (im Sinne von »unnatürlich«) klingende Muster.

Wenn man zu jedem Zeitpunkt die Werte zweier Kurvenzüge miteinander multipliziert, so spricht man von **Ringmodulation**. Dieser Vorgang hat eine Ähnlichkeit mit der Modulation einer Schwingung mit einer Hüllkurve, wie sie im SID auch vorkommt. Die Hüllkurve ist aber eine Funktion, die sich, verglichen mit dem Signal das sie moduliert, nur langsam verändert. Dadurch bleibt bei dieser Modulation der Klangcharakter des modulierten Signals erhalten, es ändert sich nur seine Lautstärke.

Bei der Ringmodulation dagegen ist das modulierende Signal von ähnlicher Beschaffenheit wie das modulierte Signal. Beide Signale dürfen ähnliche Frequenzen haben und als Kurvenzüge auch positive und negative Werte annehmen, wogegen eine Hüllkurve immer nur nicht-negative Werte hat. Bei der Ringmodulation geht im Allgemeinen der Klangcharakter beider beteiligten Kurven verloren; es entsteht ein ganz neuer Klang. Man macht sich das am besten anhand zweier Sinusschwingungen klar:

$\sin(\omega_1 t)\sin(\omega_2 t) = \frac{1}{2}(\cos(\omega_1 - \omega_2)t - \cos(\omega_1 + \omega_2)t)$

Das bedeutet, daß man durch Multiplikation zweier Sinusschwingungen ein Signalgemisch erhält, das aus Schwingungen mit der Summe und der Differenz der ursprünglichen Frequenzen besteht. Die ursprünglichen Frequenzen verschwinden dabei vollkommen. Die Formel liefert auf der rechten Seite zwar Cosinus-Terme, für den Klang ist das allerdings unerheblich, da sich Sinus und Cosinus nur durch eine Phasenverschiebung unterscheiden. Unterzieht man nun andere Kurvenformen einer Ringmodulation, so muß man alle Obertöne der einen Kurve mit allen Obertönen der anderen Kurve gemäß der obigen Formel verrechnen. Dadurch entsteht ein sehr reichhaltiges neues Obertonspektrum. Die neuen Obertöne stehen dabei nicht mehr in harmonischen, das heißt ganzzahligen Verhältnissen zueinander. Aus diesem Grund eignen sich Ringmodulatorklänge auch kaum zur Wiedergabe von Melodien. Im natürlichen Umfeld findet man unharmonische Obertonverhältnisse zum Beispiel bei Glocken, Gongs und schwingenden Metallplatten. Zu deren Nachahmung eignen sich die Ringmodulatorklänge.

Beim SID steht zur Ringmodulation nur die Dreiecksschwingung zur Verfügung. Wenn DCO1 den DCO2 moduliert, ist das Ringmodulatorprodukt nur dann hörbar, wenn DCO2 auf Dreieckskurve eingestellt ist. Noch komplexere Klänge erhält man beim SID durch Einbeziehen aller drei DCOs in die Modulationskette, wobei man durch den Signalpfad DCO3 nach DCO1 den Kreis auch noch schließen kann. Die Ergebnisse werden dann allerdings schwer vorhersagbar, lassen also noch genug Raum für Experimente und Überraschungen.

Filterung

Leider hat der SID nur ein gemeinsames Filter für die drei DCO-EG-AM-Gruppen. Möchte man unterschiedliche Klangbilder zur gleichen Zeit erzeugen, so kann man nur durch die Wahl von Kurvenform und Hüllkurve

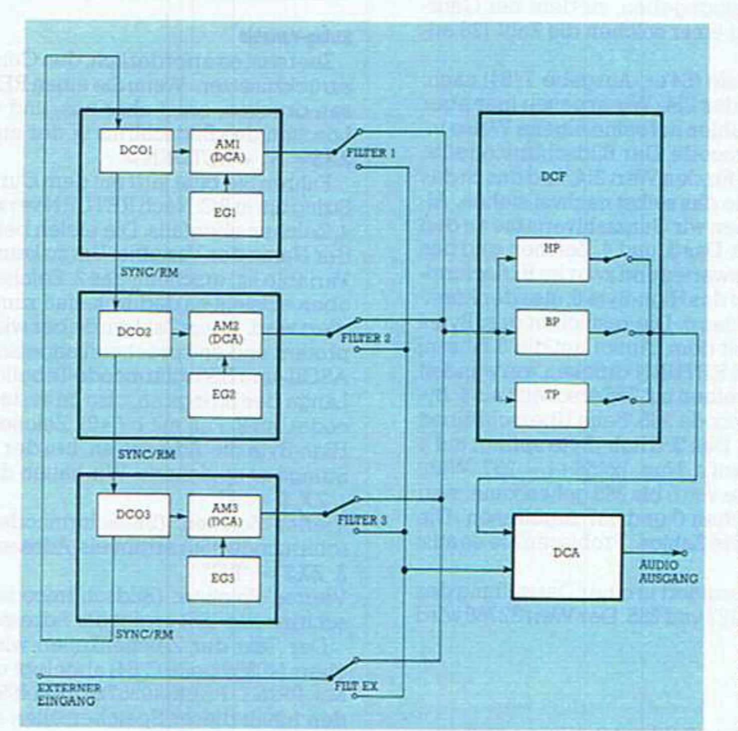


Bild 1. Blockschema des SID

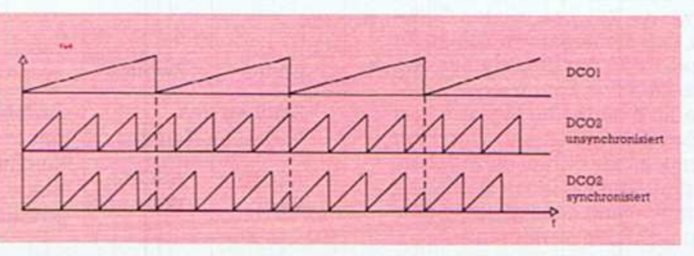


Bild 2. Die Synchronisation

Nachdem im zweiten Teil dieser Reihe einige allgemeine Grundlagen aus dem Themenkomplex »Musik und Computer« dargestellt worden sind, wird in dieser Folge ausführlich der Synthesizer Chip SID (Sound Interface Device) 6581 vorgestellt.

differenzieren. Man hat aber immerhin die Wahl, ob man die DCO-EG-AM-Produkte überhaupt durch das Filter schickt oder am Filter vorbei direkt zum Ausgang. Diese Funktion erfüllen im Blockschema die Schalter »FILTER 1«, »FILTER 2« und »FILTER 3«. Das Filter kann als Hochpaß, als Bandpaß oder als Tiefpaß wirken, wobei die Funktionen auch parallel schaltbar sind. So erhält man zum Beispiel aus der Kombination Hochpaß und Tiefpaß eine sogenannte Bandsperre (oder »Notch-Filter«), die ein schmales Band aus dem Gesamtspektrum unterdrückt. Es sind die Parameter **Eck-** beziehungsweise **Mittenfrequenz** (für HP, BP und TP gemeinsam) sowie die **Resonanz** programmierbar. Ein weiteres Qualitätskriterium eines Filters ist seine Steilheit. Diese Größe beschreibt, wie schnell ein Filter in der Umgebung der Eck- beziehungsweise Mittenfrequenz vom durchlassenden in den sperrenden Zustand übergeht.

Als Faustregel gilt, daß ein Filter um so »besser« klingt, je steiler es ist. Beim Hoch- und Tiefpaß des SID beträgt die Steilheit 12 db/Oktave, beim Bandpaß 6 dB/Oktave. 12 dB/Oktave bedeuten, daß zum Beispiel im Übergangsbereich des Tiefpasses ein Signal um das Vierfache abgeschwächt wird, wenn seine Frequenz verdoppelt wird. 6 dB/Oktave bedeuten eine Abschwächung (oder Anhebung) um das Zweifache bei Frequenzverdopplung. Die Filtersteilheit ist im Allgemeinen und auch beim SID fest vorgegeben. Zur Orientierung sei noch erwähnt, daß die Filter im professionellen Synthesizer meistens eine Steilheit von 24 dB/Oktave haben.

Die gefilterten oder ungefilterten Signale im SID werden auf einen DCA geführt, wo man noch die Gesamtamplitude des Ausgangssignals programmieren kann. Über einen Analogeingang kann man auch noch ein externes Signal gefiltert oder ungefiltert zumischen. Dieses Signal könnte zum Beispiel von einem zweiten SID stammen.

Gegenüber dem im ersten Teil vorgestellten klassischen Synthesizerkonzept findet man im SID keinen LFO, mit dem man Frequenz, Amplitude oder einen Filterparameter modulieren könnte. Diese Funktion kann man aber rein softwaremäßig realisieren. Eine Hilfe dazu können DCO3 und EG3 sein. Man kann zu jedem Zeitpunkt den Amplitudenwert von DCO3 und von EG3 abfragen. Programmiert man DCO3 als LFO, das heißt auf eine sehr niedrige Frequenz, so kann man die Amplitudenwerte von DCO3 (mit geeigneter Skalierung) zur Frequenz von DCO1 oder DCO2 addieren. Tut man das in regelmäßigen Zeitabständen und mehrfach innerhalb einer Periodendauer von DCO3, so kann man damit ein Vibrato realisieren.

Auf gleiche Weise kann man auch die von EG3 gelieferte Hüllkurve zur Modulation beispielsweise des Filters heranziehen. Beide Möglichkeiten erfordern allerdings zusätzliche schnelle Programme, die sich nur in Maschinensprache realisieren lassen.

Wenn man DCO3 und/oder EG3 zu Modulationszwecken benützt, möchte man das von AM3 produzierte Signal unter Umständen nicht hören. Dazu kann man es mit dem zusätzlichen Schalter »AUS« unterdrücken.

Im letzten Teil wurde schon angesprochen, daß ein digitaler Synthesizer wie der SID nicht durch Potentiometer und durch Spannungen beeinflußt wird, sondern durch digitalisierte Parameter. Dazu besitzt der SID 29 Register mit einer Länge von 8 Bit. Davon können 25 nur beschrieben werden (ihre Inhalte steuern das Verhalten des SID) und 4 nur gelesen werden. Die 29 Register werden durch die Adreß-Decodierungs-Hardware auf dem CPU-Speicherbereich $D400 bis $D41C abgebildet (dezimal: 54272 bis 54300).

Die Steuerung des SID

Die SID-Register lassen sich so mit allen hauptspeicherbezogenen Maschinenbefehlen oder mit PEEK und POKE ansprechen. Da in einzelnen Registern oft mehrere Bits mit unterschiedlicher Bedeutung zusammengefaßt sind, erfordert ihre Programmierung ein hohes Maß an maschinennahem Denken, egal ob in Basic oder in Maschinensprache programmiert wird.

Die Tabelle 1 zeigt die Bedeutung der einzelnen Register im Detail. Der Registersatz gliedert sich in drei mal sieben Register zur Steuerung der drei DCO-EG-AM-Gruppen für die drei Stimmen, in vier Register zur Filtersteuerung und in vier Leseregister. Die sieben Register zur Steuerung einer DCO-EG-AM-Gruppe haben für alle drei Stimmen den gleichen Aufbau und die gleiche Bedeutung.

Frequenz low/high (Register 0 und 1)

Die beiden Register 0 und 1 steuern die Frequenz von DCO1 mit einer Genauigkeit von 16 Bit. Register 0 enthält das niederwertige, Register 1 das höherwertige Byte einer 16-Bit-Größe F. Zwischen der Ausgangsfrequenz f und der Zahl F besteht der Zusammenhang:

(1) f = F x T / 2124 (Hz)

Dabei ist T die Taktfrequenz der CPU, die auch am SID anliegt. Sie beträgt bei der deutschen Version des C 64 0,985248 MHz

Damit gilt:

(2) f = F x 0,0587254 (Hz)

oder

(3) F = f x 17,0284

Die Gleichungen (2) und (3) zeigen, daß sich die Frequenzen der DCOs sehr fein, in Schritten zu zirka 1/17 Hz, programmieren lassen. Möchte man eine vorgegebene Frequenz f (im Beispiel FAUS) erzeugen, so errechnet man F nach (3),

10 FAUS = 440

Adresse	Reg.	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Registername	
54272	0	F7	F6	F5	F4	F3	F2	F1	F0	Frequenz low	Schreib-Register Stimme 1
54273	1	F15	F14	F13	F12	F11	F10	F9	F8	Frequenz high	
54274	2	P7	P6	P5	P4	P3	P2	P1	P0	Pulsweite low	
54275	3	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10	P9	P8	Pulsweite high	
54276	4	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat.	Synchronisat.	GATE	Kontrollregister	
54277	5	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2	Decay 1	Decay 0	Attack/Decay	
54278	6	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2	Release 1	Release 0	Sustain/Release	
54279	7	F7	F6	F5	F4	F3	F2	F1	F0	Frequenz low	Stimme 2
54280	8	F15	F14	F13	F12	F11	F10	F9	F8	Frequenz high	
54281	9	P7	P6	P5	P4	P3	P2	P1	P0	Pulsweite low	
54282	10	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10	P9	P8	Pulsweite high	
54283	11	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat	Synchronisat	GATE	Kontrollregister	
54284	12	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2	Decay 1	Decay 0	Attack/Decay	
54285	13	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2	Release 1	Release 0	Sustain/Release	
54286	14	F7	F6	F5	F4	F3	F2	F1	F0	Frequenz low	Stimme 3
54287	15	F15	F14	F13	F12	F11	F10	F9	F8	Frequenz high	
54288	16	P7	P6	P5	P4	P3	P2	P1	P0	Pulsweite low	
54289	17	unbenutzt	unbenutzt	unbenutzt	unbenutzt	P11	P10	P9	P8	Pulsweite high	
54290	18	Rauschen	Rechteck	Sägezahn	Dreieck	Test	Ringmodulat	Synchronisat	GATE	Kontrollregister	
54291	19	Attack 3	Attack 2	Attack 1	Attack 0	Decay 3	Decay 2	Decay 1	Decay 0	Attack/Decay	
54292	20	Sustain 3	Sustain 2	Sustain 1	Sustain 0	Release 3	Release 2	Release 1	Release 0	Sustain/Release	
54293	21	unbenutzt	unbenutzt	unbenutzt	unbenutzt	unbenutzt	GF 2	GF 1	GF 0	Grenzfrequenz low	Filter
54294	22	GF 10	GF 9	GF 8	GF 7	GF 6	GF 5	GF 4	GF 3	Grenzfrequenz high	
54295	23	Resonanz 3	Resonanz 2	Resonanz 1	Resonanz 0	Filtex	Filter 3	Filter 2	Filter 1	Resonanz/Filter	
54296	24	Aus	Hochpass	Bandpass	Tiefpass	L 3	L 2	L 1	L 0	Mode/Lautstärke	
54297	25	Pot X 7	Pot X 6	Pot X 5	Pot X 4	Pot X 3	Pot X 2	Pot X 1	Pot X 0	Potentiometer X	Lese-Register
54298	26	Pot Y 7	Pot Y 6	Pot Y 5	Pot Y 4	Pot Y 3	Pot Y 2	Pot Y 1	Pot Y 0	Potentiometer Y	
54299	27	O7	O6	O5	O4	O3	O2	O1	O0	Oszillator 3	
54300	28	H7	H6	H5	H4	H3	H2	H1	H0	Hüllkurve Osz 3	

Pulsbreite hier 50%

Tabelle 1. Die Register des SID

Wert Dez.	Hex	Attack	Decay/Release
0	0	2 ms	6 ms
1	1	8 ms	24 ms
2	2	16 ms	48 ms
3	3	24 ms	72 ms
4	4	38 ms	114 ms
5	5	56 ms	168 ms
6	6	68 ms	204 ms
7	7	80 ms	240 ms
8	8	100 ms	300 ms
9	9	250 ms	750 ms
10	A	500 ms	1,5 s
11	B	800 ms	2,4 s
12	C	1 s	3 s
13	D	3 s	9 s
14	E	5 s	15 s
15	F	8 s	24 s

Die Zeiten gelten bei einer System-Taktfrequenz von 1 MHz. Da diese Frequenz beim deutschen C 64 mit 0,985248 MHz nur gering davon abweicht, ist die Tabelle auch für diese Frequenz brauchbar.
Quelle: Commodore 64 Programmers Reference Guide

Tabelle 2. Hüllkurvengeschwindigkeiten

20 F = FAUS * 17.0284
man zerlegt F in das Low- und das High-Byte,
30 F = INT(F+0.5) :REM RUNDUNG
40 HI = INT(F/256)
50 LO = F − 256*HI
und besetzt Register 0 und 1 damit
60 SID = 54272 :REM BASISADRESSE
70 POKE SID,LO
80 POKE SID+1,HI

Pulsweite low/high (Register 2 und 3)

Der Inhalt dieser Register steuert das Tastverhältnis der Rechteckkurve. Er ist nur dann bedeutend, wenn als Kurvenform das Rechteck gewählt wird. Die Pulsweite kann auf 12 Bit genau festgelegt werden. Register 2 enthält das Low-Byte, Register 3 das High-Byte, von dem nur die unteren 4 Bits (P8 bis P11) berücksichtigt werden. Zwischen der 12-Bit-Größe P und dem Tastverhältnis PAUS besteht der Zusammenhang:
(4) PAUS = P / 40.95 (%)
oder
(5) P = PAUS * 40.95
Die Programmierung gestaltet sich dann in der Praxis analog zu der der Frequenz:
90 PAUS = 50
100 P = PAUS * 40.95
110 P = INT(P+0.5) :REM RUNDUNG
120 HI = INT (P/256)
130 LO = P − 256*HI
140 POKE SID+2,LO
150 POKE SID+3,HI

Kontrollregister (Register 4)

Jedes Bit dieses Registers hat eine eigene Bedeutung.
GATE (Bit 0)

Dieses Bit steuert den Hüllkurvengenerator EG1. Wird es gesetzt, startet EG1 eine Attack-Decay-Sequenz. Die Hüllkurve bleibt anschließend auf dem programmierten Sustain-Pegel, bis das GATE-Bit zurückgesetzt wird. Durch das Zurücksetzen geht die Hüllkurve in die Release-(Auskling-)Phase.

Synchronisation (Bit 1)

Wird es gesetzt, so wird DCO1 von DCO3, wie schon beschrieben, synchronisiert.

Ringmodulation (Bid 2)

Wird es gesetzt, so erzeugt DCO1 das Ringmodulatorprodukt der Dreieckskurven von DCO3 und DCO1. Diese wird allerdings nur dann hörbar, wenn als Kurvenform von DCO1 das Dreieck gewählt wird.

Test (Bit 3)

Bei gesetztem Test-Bit wird DCO1 auf Nullpegel gezwungen. Er erzeugt in diesem Zustand keine Schwingung. Man kann DCO1 damit softwaregesteuert synchronisieren, um, ähnlich wie durch die Synchronisation durch DCO3, komplexere Kurvenformen zu erhalten.

Kurvenform (Bit 4 bis 7)

Durch Setzen eines dieser 4 Bits wählt man eine der Kurvenformen Dreieck, Sägezahn, Rechteck oder Rauschen. Es muß mindestens eines dieser Bits gesetzt sein, damit überhaupt etwas hörbar wird. Eine Eigenart des SID ist es, daß sich die Kurvenformen **nicht** additiv verhalten. Werden mehrere der Bits 4 bis 7 zugleich gesetzt, so erzeugt der SID eine Kurvenform, deren Amplitudenwerte man durch logische AND-Verknüpfung der Amplitudenwerte der einzelnen Kurven erhält. Diese AND-Verknüpfung muß man sich bitweise auf die Amplitudenwerte darstellenden Bytes angewandt vorstellen.

Das **Rauschen** verdient noch eine besondere Betrachtung. Bei Rauschen kann man nicht von einer Frequenz im üblichen Sinne reden. Dennoch ist der Charakter des SID-Rauschens über die Größe F in Register 0 und 1 beeinflußbar. Rauschen wird im SID durch eine Quasi-Zufallsfolge von numerischen Werten realisiert. Die Rate, mit der der DCO diese Zufallszahlen erzeugt, ist genau die durch F programmierte Frequenz. Ein Rau-

```
100 SI=54272     :REM BASISADRESSE                <248>
110 :                                              <168>
120 REM FREQUENZ                                   <119>
130 READ FAUS :F=INT(FAUS*17.0284+0.5)             <236>
140 HI=INT(F/256) :LO=F-256*HI                     <183>
150 POKE SI,LO :POKE SI+1,HI                       <149>
160 :                                              <218>
170 REM PULSWEITE (TASTVERHAELTNIS)                <205>
180 READ PAUS :P=INT(PAUS*40.95+0.5)               <216>
190 HI=INT(P/256) :LO=P-256*HI                     <253>
200 POKE SI+2,LO :POKE SI+3,HI                     <165>
210 :                                              <012>
220 REM WELLENFORM                                 <102>
230 READ WF:WF=WF AND 254 :REM GATE AUS            <077>
240 :                                              <042>
250 REM A D S R                                    <179>
260 READ A:READ D:POKE SI+5,16*A+D                 <140>
270 READ S:READ R:POKE SI+6,16*S+R                 <215>
280 :                                              <083>
290 REM FILTER AUS, LAUTSTAERKE MAX                <184>
300 POKE SI+23,0:POKE SI+24,15                     <218>
310 :                                              <113>
320 REM AUF TASTENDRUCK WARTEN                     <197>
330 REM UND ADSR SEQUENZ AUSLOESEN                 <197>
340 READ T                                         <048>
350 GET A$:IF A$="" THEN 350                       <196>
360 POKE SI+4,WF OR 1                              <036>
370 FOR I=1 TO T:NEXT                              <212>
380 POKE SI+4,WF:GOTO 350                          <178>
500 REM================================           <036>
510 REM KLANGPARAMETER                             <156>
520 REM================================           <057>
530 DATA 440      :REM FREQUENZ (HZ)               <091>
540 DATA 50       :REM TASTVERHAELTNIS (%)         <198>
550 DATA 64       :REM KURVENFORM                  <237>
560 DATA 2        :REM ATTACK                      <104>
570 DATA 8        :REM DECAY                       <038>
580 DATA 0        :REM SUSTAIN                     <233>
590 DATA 8        :REM RELEASE                     <213>
600 DATA 150      :REM ZEIT ZWISCHEN               <227>
610 REM                ATTACK UND RELEASE          <147>
```

Listing 1. Mit Programm 1 kann man sich damit vertraut machen, wie einzelne Töne mit verschiedenen Kurvenformen und Hüllkurven klingen. Die Parameter lassen sich in den DATA-Zeilen am Ende des Programms leicht modifizieren.

```
10 REM================================            <118>
20 REM PROGRAMM 2                                  <058>
30 REM                                             <173>
40 REM EINZEILER FUER SIGNALTON                    <063>
50 REM================================            <158>
60 S=54272:FOR I=0 TO 6:READ X:POKE S+I,X:NEXT
   :POKE S+24,15:POKE S+4,17:POKE S+4,16
   :DATA 0,99,0,8,0,11,11                          <041>
```

Listing 2. Dieser Einzeiler ist gegenüber dem übersichtlichen, aber etwas aufgeblasenen Programm 1 so ziemlich die kompakteste Lösung, um dem SID einen Ton zu entlocken. Zur Eingabe muß man die Basic-Schlüsselwörter abkürzen.

```
100 SI=54272    :REM BASISADRESSE                  <248>
110 :                                              <168>
120 REM TASTVERHAELTNIS = 50%                      <079>
130 POKE SI+2,0 :POKE SI+3,8                       <155>
140 :                                              <198>
150 REM ADSR                                       <079>
160 A=0  :D=0   :POKE SI+5,16*A+D                  <221>
170 S=15 :R=0   :POKE SI+6,16*S+R                  <094>
180 :                                              <238>
190 REM FILTER AUS UND LAUTSTAERKE MAX.            <060>
200 POKE SI+23,0:POKE SI+24,15                     <117>
210 :                                              <012>
220 REM FLO=0, KURVENFORM                          <244>
230 POKE SI,0                                      <117>
240 POKE SI+4,65                                   <152>
250 :                                              <052>
260 REM FHI AUF- UND ABWAERTS STEUERN              <218>
270 F0=50 :F1=100                                  <144>
280 FOR I=0.4 TO 50 STEP .2                        <057>
290 :  FOR F=F0 TO F1 STEP I                       <089>
300 :     POKE SI+1,F                              <231>
310 : NEXT F                                       <057>
320 :  FOR F=F1 TO F0 STEP -I                      <034>
330 :     POKE SI+1,F                              <005>
340 :  NEXT F                                      <087>
350 NEXT I                                         <042>
360 POKE SI+4,64                                   <016>
```

Listing 3. Durch Verändern des höherwertigen Frequenz-Bytes in kleinen Schritten entsteht der Eindruck, daß der Ton kontinuierlich in der Höhe schwankt

```
134 REM DAS MASCHINENPROGRAMM BELEGT      <155>
136 REM DEN BEREICH ($9000-$90AC)         <092>
140 REM                                   <027>
150 REM AUFRUFBEDINGUNGEN                 <020>
160 REM                                   <047>
170 REM SYS PAR,S,PW,K,A,D,S,R            <190>
180 REM                                   <067>
190 REM      BELEGT STIMME S (1,2,3) MIT  <075>
200 REM                                   <087>
210 REM      PW (PULSWEITE 0...4095)      <167>
220 REM      K  (KURVENFORM,SYNCHRONISA-  <151>
230 REM          TION,RINGMODULATION)     <048>
240 REM      A,D,S,R (HUELLKURVE 0..15)   <119>
250 REM                                   <137>
260 REM SYS EIN,S,F                       <096>
270 REM                                   <158>
280 REM      SCHALTET STIMME S MIT        <012>
290 REM      FREQUENZ F (0..65535) EIN    <041>
300 REM                                   <188>
310 REM SYS AUS,S                         <045>
320 REM                                   <208>
330 REM      SCHALTET STIMME S AUS        <061>
340 REM=============================      <132>
360 :                                     <163>
370 A=36864:B=37036                       <162>
380 FOR I=A TO B                          <032>
390 READ X:POKE I,X:S=S+X:NEXT I          <157>
400 IF S<>17579 THEN PRINT"TIPPFEHLER"    <087>
410 :                                     <213>
420 PAR=A+66  :EIN=A+122 :AUS=A+161       <049>
430 :                                     <233>
440 REM DATAS FUER MASCHINENPROGRAMM      <226>
450 :                                     <253>
460 DATA 000,000,000,000,000,032,253,174  <031>
470 DATA 032,158,183,224,004,048,003,076  <085>
480 DATA 072,178,224,000,240,249,202,188  <099>
490 DATA 027,144,096,000,007,014,152,072  <096>
500 DATA 032,253,174,032,158,183,224,016  <117>
510 DATA 048,003,076,072,178,104,168,138  <139>
520 DATA 096,032,030,144,010,010,010,010  <101>
530 DATA 141,000,144,032,030,144,013,000  <111>
540 DATA 144,096,032,005,144,138,072,152  <157>
550 DATA 072,032,253,174,032,235,183,142  <165>
560 DATA 000,144,104,168,104,170,173,000  <158>
570 DATA 144,157,002,144,165,021,201,022  <170>
580 DATA 048,003,076,072,178,153,003,212  <195>
590 DATA 165,020,153,002,212,032,049,144  <192>
600 DATA 153,005,212,032,049,144,153,006  <206>
610 DATA 212,096,032,005,144,138,072,152  <223>
620 DATA 072,032,253,174,032,138,173,032  <234>
630 DATA 247,183,104,168,165,020,153,000  <243>
640 DATA 212,165,021,153,001,212,104,170  <233>
650 DATA 189,002,144,009,001,153,004,212  <252>
660 DATA 096,032,005,144,189,002,144,041  <017>
670 DATA 254,153,004,212,096              <207>
```

Listing 4. Die Programmierung des SID über POKE-Befehle ist mühsam. Drei kleine Maschinenprogramme, die man auch als Erweiterung des BASIC-Interpreters auffassen kann, werden von Programm 5 und 6 aufgerufen.

```
100 SI=54272   :REM BASISADRESSE             <248>
110 PAR=36930:EIN=36986:AUS=37025            <182>
120 :                                        <178>
130 REM FILTER AUS UND LAUTSTAERKE MAX.      <000>
140 POKE SI+23,0:POKE SI+24,15               <057>
150 :                                        <208>
160 REM PARAMETERSAETZE ABARBEITEN           <105>
170 READ N                                   <127>
180 FOR I=1 TO N                             <083>
190 :  READ F,A,D,S,R,T1,T2                  <002>
200 :  SYS PAR,1,0,128,A,D,S,R               <221>
210 :  SYS EIN,1,F                           <085>
220 :  PRINT"BEISPIEL ";I;" GATE AN";        <046>
230 :  FOR J=1 TO T1:NEXT J                  <253>
240 :  SYS AUS,1                             <014>
250 :  PRINT"(2SPACE)GATE AUS"               <027>
260 :  FOR J=1 TO T2:NEXT J                  <029>
270 NEXT I                                   <218>
500 DATA 6 :REM ANZAHL DER P.SAETZE          <090>
510 REM=============================         <046>
520 REM PARAMETERSAETZE                      <006>
530 REM                                      <163>
540 REM        F   A  D  S  R    T1   T2     <040>
550 REM=============================         <087>
560 DATA 50000,  0, 0,15, 0, 1500, 200       <000>
570 DATA 25000,  0,12, 0,12,  500, 500       <014>
580 DATA  5000,  8, 5, 2, 5, 1500, 500       <197>
590 DATA  1000,  0,13, 0,12,  500,1500       <030>
600 DATA   750, 14, 0,15,13, 2000,2500       <105>
610 DATA    80,  8, 8, 8,10, 2500, 500       <188>
```

Listing 5. Programm 5 zeigt, wie vielfältig Rauschen klingen kann. Die Parametersätze in den DATA-Zeilen kann man beliebig verändern oder erweitern.

```
100 DIM F(24)  :REM ARRAY F. FREQUENZEN     <166>
110 SID=54272  :REM BASISADRESSE            <070>
120 PAR=36930:EIN=36986:AUS=37025           <192>
130 :                                       <188>
140 REM TONLEITER-FREQUENZEN BERECHNEN      <139>
150 FAUS=110:H=2↑(1/12)                     <175>
160 FOR I=0 TO 24                           <086>
170 :  F(I)=INT(FAUS*17.0284+0.5)           <248>
180 :  FAUS=FAUS*H                          <242>
190 NEXT I                                  <137>
200 :                                       <002>
210 REM PARAMETER FESTLEGEN                 <159>
220 PW=2048          :REM PULSWEITE         <142>
230 K =32            :REM KURVENFORM        <032>
240 A=0: D=9: S=0 :R=9                      <098>
250 SYS PAR,1,PW,K,A,D,S,R                  <252>
260 SYS PAR,2,PW,K,A,D,S,R                  <008>
270 SYS PAR,3,PW,K,A,D,S,R                  <019>
280 :                                       <083>
290 REM FILTER AUS UND LAUTSTAERKE MAX.     <161>
300 POKE SI+23,0:POKE SI+24,15              <218>
310 :                                       <113>
320 REM ZUFALLSMELODIE                      <240>
330 I=1                                     <119>
340 N=INT(RND(1)*25)                        <171>
350 SYS EIN,I,F(N)                          <095>
360 FOR J=0 TO 20:NEXT J                    <034>
370 SYS AUS,I                               <111>
380 FOR J=0 TO 20:NEXT J                    <054>
390 I=I+1:IF I=4 THEN I=1                   <109>
400 GOTO 340                                <177>
```

Listing 6. Die Frequenzen der temperierten Stimmung werden über 2 Oktaven errechnet. Anschließend wird daraus eine Zufallstonfolge erzeugt. Durch zyklisches Ansteuern aller drei Stimmen haben die Töne Zeit zum Ausklingen.

schen mit »hoher Frequenz« klingt heller oder »weißer« als Rauschen mit niedriger Frequenz.

Bei der Programmierung des Kontrollregisters muß man sich vorher den Wert jedes einzelnen Bits zurechtlegen und in das Byte packen. Das Anstoßen eines Rechteckklanges beispielsweise wird durch Setzen von Bit 0 und Bit 6 erreicht. Der numerische Wert des Kontrollbytes ist dann: $2^0+2^6=65$

also: 240 POKE SID+4,65

Ein hörbares Resultat wird aber auch erst dann erzielt, wenn vorher die Hüllkurvenparameter vernünftig gesetzt sind.

ADSR (Register 5 und 6)

Die Parameter A, D, S und R können in 16 Abstufungen entsprechend 4 Bit Auflösung programmiert werden. Die Stufung für den Sustain-Pegel ist linear. S=0 entspricht dem Ruhepegel, S=15 entspricht dem Maximalpegel nach Attack. Bei S=15 besitzt die Hüllkurve keine Decay-Phase. Die Abstufungen für die Attack-, Decay- und Release-Zeiten sind vernünftigerweise nicht linear, um sehr kurze und sehr lange Zeiten gleichermaßen zu ermöglichen. Tabelle 2 enthält die realisierbaren Attack-, Decay- und Release-Zeiten.

Decay und Release können direkt in die Register 5 und 6 geschrieben werden, während Attack und Sustain vorher mit 16 zu multiplizieren sind, was einer Linksverschiebung um vier binäre Stellen entspricht. Ein Beispiel:

```
160 A =  2     :REM 16 MS
170 D = 12     :REM 3 S
180 S =  1
190 R = 10     :REM 1.5 S
200 POKE SID+5,A*16+D
210 POKE SID+6,S*16+R
```

Nach vorläufigem Umgehen des Filters und Setzen der maximalen Lautstärke

```
220 POKE SID+23,0 :REM FILTER AUS
230 POKE SID+24,15 :REM LAUTSTÄRKE
```

führt das Setzen des GATE-Bits:

```
240 POKE SID+4,65 :REM GATE AN
```

zu einem vernünftigen Resultat. Zum Abschalten des Klanges kann man einfach das Kontrollregister mit 0 besetzen,

```
270 POKE SID+4,0
```

doch dann hat DCO1 keine Gelegenheit, der Release-Phase entsprechend auszuklingen, da auch das Kurvenform-Bit (Nr. 6) zurückgesetzt ist. Besser ist also, Bit 6 gesetzt zu lassen:

```
250 GET A$:IF A$="" THEN 160
260 REM WARTE AUF EINE TASTE
270 POKE SID+4,64 :REM GATE AUS
```

Die Basic-Zeilen zeigen natürlich keinen effizienten Programmierstil; sie sollen nur die Vorgehensweise darstellen.

Auf die Steuerung des Filters und auf die Anwendung der Leseregister wird im nächsten Teil noch ausführlich eingegangen werden. So ist die Abfrage eines analogen Gebers (zum Beispiel Paddle) erst durch den SID möglich. Die sechs kleinen Beispielprogramme demonstrieren einige Effekte, die man mit den bis hierher beschriebenen Registern realisieren kann.

(Thomas Krätzig/aa)

500 Mark für das lustigste Programm

Der Gewinner des Programmierwettbewerbs »Das lustigste Programm« steht fest. Die »Notlandung« von Rainer Schleeweiß hat eine gelungene Pointe, die auch nach wiederholtem Anschauen immer wieder zum Lachen reizt. Die Wirkung wird jedoch nicht nur durch die Idee alleine hervorgerufen, sondern auch durch die gelungene grafische Umsetzung.


Rainer Schleeweiß, der Autor der »Notlandung«

NOTLANDUNG



Ein »lustiges Programm« zu schreiben ist nicht so einfach. Wie bei Witzen im allgemeinen wird der Lacherfolg nicht alleine durch den Inhalt erreicht, sondern vor allem durch die richtige Plazierung der Pointe und nicht zuletzt durch den Erzähler.

Die Notlandung ist ein »programmierter Gag«, eine Art Zeichentrickfilm, kein Witz, den man erzählen könnte. Der Titel des Programms könnte auch treffender nicht sein, und er hat nichts mit materiellem Schaden zu tun, sondern mehr mit einer menschlichen Schwäche. Doch mehr wird nicht verraten. Wir mußten einige REM-Zeilen des sonst gut dokumentierten Programms entfernen, um Ihnen nicht schon bei der Eingabe oder beim Studieren des Programmlistings einen Teil des Spaßes vorwegzunehmen. Wir hoffen, daß auch Sie mit unserer Wahl einverstanden sind und genauso lachen wie wir. (gk)

»Notlandung«, das lustige Programm

```
1 REM *************************          <128>
2 REM *                       *          <229>
3 REM *     'NOTLANDUNG'      *          <046>
4 REM *                       *          <231>
5 REM *   VON R.SCHLEEWEISS   *          <154>
6 REM *     6094 WEINHEIM     *          <018>
7 REM *************************          <134>
8 REM                                    <151>
10 REM *SPRITE-REGISTER AUF 0 SETZEN*    <059>
15 FOR X=53248 TO 53263:POKE X,0:NEXT X  <224>
20 GOSUB 2000:REM ***TITEL***            <170>
23 POKE 53265,PEEK(53265)AND 239
   :REM BILDSCHIRM WEGBLENDEN            <201>
25 GOSUB 3000:REM ***LANDSCHAFT***       <006>
30 V=53248:S=54272                       <111>
35 REM *** SONNE ***                     <049>
40 FOR I8=12736 TO 12798:READ Q8:POKE I8,Q8:NEXT
                  <155>
45 POKE 2047,199:REM ZEIGER              <239>
50 POKE V+21,128:REM SPR.EINSCH.         <199>
60 POKE V+27,128:REM HINTERGR.PRIORIT.   <180>
65 POKE V+46,8  :REM FARBE               <207>
70 POKE V+14,72:REM X-POS.               <069>
75 POKE V+15,98:REM Y-POS.               <084>
77 FOR BR=1 TO 350:NEXT BR               <209>
78 POKE 53265,PEEK(53265)OR 16
   :REM BILDSCHIRM ZURUECKSETZEN         <241>
80 REM **HELIKOPTER-LANDUNG**            <180>
90 FOR I=12288 TO 12350:READ Q:POKE I,Q:NEXT
                  <223>
95 FOR I6=12608 TO 12670:READ Q6:POKE I6,Q6
   :NEXT I6                              <060>
100 FOR I7=12672 TO 12734:READ Q7:POKE I7,Q7
   :NEXT I7                              <072>
110 POKE 2040,192:REM ZEIGER SETZEN      <251>
120 POKE V+21,129:REM SPRITE EINSCHALTEN <008>
130 POKE V+29,1:REM IN X-RICHT.VERGR.    <034>
140 POKE V+39,8:REM SPRITE-FARBE         <032>
150 POKE V+28,1:REM FARBMODUS SETZEN     <057>
160 POKE V+37,1:REM 1.FARBE              <134>
170 POKE V+38,3:REM 2.FARBE              <148>
180 POKE V,150:REM SPRITE X-POS.         <168>
181 FOR L=S TO S+24:POKE L,0:NEXT L      <015>
182 FOR K=7 TO 24:READ X:POKE S+K,X:NEXT K <050>
183 POKE S+11,65:POKE S+18,65            <018>
184 POKE S+5,9:POKE S+6,2                <072>
190 FOR I=35 TO 157                      <227>
191 POKE S+5,0                           <228>
193 POKE S+1,6 :REM TONHOEHE             <011>
194 POKE S,177:REM NIED.FREQUENZ         <254>
196 POKE S+4,65 :REM RECHTECK            <053>
198 POKE S+4,32 :REM RECHTECK AUSLOESEN  <224>
200 POKE V+1,I:REM SPRITE Y-POS.         <075>
210 POKE 12294,5:POKE 12295,81:POKE 12296,84
   :REM ROTOR-DREH-EFFEKT M.ZEILE 230    <188>
220 FOR BR=1 TO 5 :NEXT BR               <253>
230 POKE 12294,21:POKE 12295,85:POKE 12296,85
                  <233>
240 NEXT I                               <187>
250 REM ***ROTOR LAEUFT NACH***          <246>
260 POKE V,150:POKE V+1,157              <127>
265 POKE S+1,4:POKE S,177                <030>
270 FOR I=1 TO 15                        <198>
275 POKE S+5,0:POKE S+4,65:POKE S+4,32   <101>
280 POKE 12294,5:POKE 12295,81:POKE 12296,84
                  <233>
290 FOR BR=1 TO 65:NEXT BR               <122>
300 POKE 12294,21:POKE 12295,85:POKE 12296,85
                  <048>
310 NEXT I                               <002>
315 POKE S+24,22                         <198>
320 REM ***MANN NACH RECHTS***           <217>
325 POKE S,220:POKE S+5,0 :POKE S+6,180  <024>
330 FOR I1=12352 TO 12414:READ Q1:POKE I1,Q1
   :NEXT                                 <141>
340 FOR I2=12416 TO 12478:READ Q2:POKE I2,Q2
   :NEXT                                 <166>
350 POKE V+21,131                        <026>
360 POKE V+40,0                          <192>
370 POKE V+3,157                         <006>
380 P=193                                <028>
390 FOR I=165 TO 219 STEP 3              <187>
```

```
400 POKE V+2,I                                <207>
410 POKE 2041,P:POKE S+1,R:POKE S+4,65
 :POKE S+4,64                                 <035>
420 P=P+1:IF P>194 THEN P=193                 <124>
430 IF I=219 THEN 450                         <017>
440 FOR BR=1 TO 50:NEXT BR                    <010>
450 NEXT I                                    <142>
460 :                                         <007>
470 FOR I1=12352 TO 12414:READ Q1:POKE I1,Q1
 :NEXT                                        <025>
480 POKE 2041,193                             <008>
490 POKE V+21,131:REM SPR.1,3 EINSCH.         <010>
500 POKE V+40,0                               <076>
510 POKE V+3,157:REM Y-POS.                   <001>
520 POKE V+2,219:REM X-POS.                   <009>
530 FOR BR=1 TO 150:NEXT BR                   <150>
540 :                                         <088>
550 FOR I3=12480 TO 12542:READ Q3:POKE I3,Q3
 :NEXT                                        <118>
560 POKE 2043,195                             <093>
570 POKE V+21,139:REM SPR.1,3,4 EINSCH.       <195>
580 POKE V+42,7                               <166>
590 POKE V+7,168:REM Y-POS.                   <088>
600 POKE V+6,238:REM X-POS.                   <094>
610 :                                         <158>
620 POKE 2045,197                             <157>
630 POKE V+21,171                             <055>
640 POKE V+44,7                               <228>
650 POKE V+11,180:REM Y-POS.                  <185>
660 POKE V+10,249:REM X-POS.                  <199>
670 P=197                                     <067>
675 POKE S+6,200:POKE S+5,0                   <140>
680 FOR I=1 TO 80                             <099>
685 POKE S,9:POKE S+1,I:POKE S+4,129          <060>
690 POKE 2045,P                               <146>
700 P=P+1:IF P>198 THEN P=197                 <157>
710 FOR BR=1 TO 10:NEXT BR                    <021>
720 NEXT I:POKE S+4,0                         <251>
730 FOR Y=12480 TO 12542:POKE Y,0:NEXT Y      <164>
740 POKE V+21,PEEK(V+21)AND 255-40            <220>
755 POKE S,220:POKE S+5,0:POKE S+6,180        <199>
760 FOR I1=12352 TO 12414:READ Q1:POKE I1,Q1
 :NEXT                                        <060>
770 FOR I2=12416 TO 12478:READ Q2:POKE I2,Q2
 :NEXT                                        <086>
780 POKE V+21,131                             <202>
790 POKE V+40,0                               <112>
800 POKE V+3,157                              <150>
810 P=193                                     <204>
820 FOR I=219 TO 165 STEP-3                   <022>
830 POKE V+2,I                                <127>
840 POKE 2041,P:POKE S+1,R:POKE S+4,65
 :POKE S+4,64                                 <211>
850 P=P+1:IF P>194 THEN P=193                 <044>
860 FOR BR=1 TO 50:NEXT BR                    <176>
870 NEXT I                                    <052>
880 REM ***FERTIG ZUM ABFLUG***               <108>
890 POKE 2040,192:REM ZEIGER SETZEN           <010>
900 POKE V+21,129:REM SPR.1 EINSCH.           <078>
910 FOR BR=1 TO 800:NEXT BR                   <021>
920 REM *ROTOR BEGINNT ZU DREHEN*             <128>
930 POKE V,150:POKE V+1,157                   <031>
935 POKE S+1,4:POKE S,177:POKE S+24,31
 :POKE S+6,2                                  <228>
940 FOR I=1 TO 15                             <102>
945 POKE S+5,0:POKE S+4,65:POKE S+4,32        <005>
950 POKE 12294,5:POKE 12295,81:POKE 12296,84
                       <137>
960 FOR BR=1 TO 50:NEXT BR                    <020>
970 POKE 12294,21:POKE 12295,85:POKE 12296,85
                       <208>
980 NEXT I                                    <162>
990 REM ***HELIKOPTER-START***                <030>
995 POKE S+1,6:POKE S,177                     <252>
1000 FOR I=157 TO 52 STEP-1                   <148>
1005 POKE S+5,0:POKE S+4,65:POKE S+4,32       <065>
1010 POKE V,307-I :REM X-POS.                 <010>
1020 POKE V+1,I:REM Y-POS.                    <171>
1030 POKE 12294,5:POKE 12295,81:POKE 12296,84
 :REM ROTOR-DREH-EFFEKT M.ZEILE 1040          <036>
1040 POKE 12294,21:POKE 12295,85:POKE 12296,85
                       <023>
1050 NEXT I                                   <233>
1060 REM ***RECHTE X-POS.***                  <019>
1065 POKE S+1,6:POKE S,177                    <067>
1070 FOR Y=50 TO 80                           <047>
1072 IF Y<65 THEN 1075                        <170>
1073 POKE S+24,96-Y:REM MOTOR LEISER          <236>
1075 POKE S+5,0:POKE S+4,65:POKE S+4,32       <136>
1080 POKE V+16,1                              <151>
1090 POKE V+0,Y-50:REM X-POS.                 <016>
1100 POKE V+1,101-Y:REM Y-POS.                <073>
1110 POKE 12294,5:POKE 12295,81:POKE 12296,84
 :REM ROTOR-DREHEFFEKT M.ZEILE 1120           <070>
1120 POKE 12294,21:POKE 12295,85:POKE 12296,85
                 <103>
1130 NEXT Y                                   <073>
1135 POKE V+16,0:POKE V+0,0                   <042>
1140 REM *** END-ZEICHEN 1.TEIL ***           <154>
1141 FOR I=12288 TO 12350:READ Q:POKE I,Q:NEXT I
                 <071>
1142 FOR I1=12352 TO 12414:READ Q1:POKE I1,Q1
 :NEXT I1                                     <054>
1143 FOR I2=12416 TO 12478:READ Q2:POKE I2,Q2
 :NEXT I2                                     <071>
1145 REM *** SONNE NACH UNTEN ***             <047>
1146 FOR I=S TO S+24:POKE I,0:NEXT            <134>
1147 POKE S+24,10:POKE S+5,10:POKE S+6,100
 :POKE S+3,100:POKE S+2,100                   <210>
1150 FOR I=72 TO 39 STEP-1                    <252>
1152 READ HF,LF,DR                            <025>
1154 POKE S+1,HF:POKE S,LF                    <231>
1155 POKE S+4,65                              <230>
1160 POKE V+14,I:POKE V+15,170-I              <236>
1170 FOR BR=1 TO DR:NEXT BR                   <024>
1172 POKE S+4,33                              <242>
1180 NEXT I                                   <107>
1182 FOR I=S TO S+24:POKE I,0:NEXT            <170>
1190 REM *SPRITE-REGISTER AUF 0 SETZEN*       <219>
1200 FOR X=53248 TO 53263:POKE X,0:NEXT X     <133>
1204 REM                                      <071>
1205 REM *** END-ZEICHEN 2.TEIL ***             MI
 T ZEILEN 1141,1142,1143                      <061>
1206 REM                                      <073>
1208 FOR I3=12480 TO 12542:READ Q3:POKE I3,Q3
 :NEXT I3                                     <134>
1210 FOR I4=12544 TO 12606:READ Q4:POKE I4,Q4
 :NEXT I4                                     <143>
1220 FOR I5=12608 TO 12670:READ Q5:POKE I5,Q5
 :NEXT I5                                     <160>
1230 FOR I6=12672 TO 12734:READ Q6:POKE I6,Q6
 :NEXT I6                                     <177>
1235 POKE V+28,0:REM MEHRFARBMOD.AUF 0        <094>
1240 POKE V+39,1                              <060>
1250 POKE V+0,150                             <111>
1260 POKE V+1,110                             <118>
1270 POKE V+29,1                              <089>
1290 FOR J=0 TO 6                             <150>
1300 POKE V+21,1                              <112>
1310 POKE 2040,192+J                          <060>
1320 FOR BR=1 TO 30:NEXT BR                   <124>
1330 NEXT J                                   <003>
1340 FOR BR=1 TO 1000:NEXT BR                 <238>
1350 FOR K=6 TO 0 STEP-1                      <088>
1360 POKE 2040,192+K                          <111>
1370 FOR BR=1 TO 30:NEXT BR                   <174>
1380 NEXT K                                   <054>
1390 POKE V+21,0                              <201>
1400 FOR BR=1 TO 400:NEXT BR                  <253>
1410 GOTO 1290                                <220>
1500 END                                      <097>
2000 PRINT"{CLR}":REM ***TITEL***             <142>
2010 POKE 53280,2 :POKE 53281,6               <014>
2012 FOR BR=1 TO 800:NEXT BR                  <103>
2015 S=54272                                  <239>
2017 FOR I=S TO S+24:POKE I,0:NEXT I          <057>
2018 POKE S+24,10:POKE S+5,12*16+0
 :POKE S+6,15*16+10                           <163>
2020 FOR A=0 TO 39                            <159>
2025 POKE S+1,A+15:POKE S,226:POKE S+4,33     <176>
2030 POKE 1024+A+40*16,64                     <245>
2040 POKE 55296+A+40*16,7                     <016>
2050 NEXT A                                   <205>
2060 FOR I=0 TO 65                            <207>
2065 POKE S+1,65-I:POKE S,226:POKE S+4,129    <029>
2067 IF I>24 THEN 2090                        <120>
2070 POKE 1024+10+40*I,66                     <034>
2080 POKE 55296+10+40*I,1                     <021>
2090 NEXT I                                   <253>
```

»Notlandung«, das lustige Programm

»Notlandung«, das lustige Programm (Fortsetzung)

```
2095 POKE S+4,0                                  <091>
2100 FOR BR=1 TO 500:NEXT BR                     <189>
2110 PRINT"{7DOWN,18RIGHT,CYAN}'NOTLANDUNG'"
                         <139>
2120 FOR BR=1 TO 700:NEXT BR                     <211>
2130 PRINT"{10DOWN,15RIGHT,GREY 3,3SPACE}VON
   R.SCHLEEWEISS"                                <225>
2132 FOR BR=1 TO 500:NEXT BR                     <221>
2135 PRINT"{DOWN,15RIGHT,ORANGE,4SPACE}MUSIC
   VON HARDY"                                    <109>
2140 FOR BR=1 TO 2000:NEXT BR                    <018>
2150 RETURN                                      <252>
3000 PRINT"{CLR}":REM ***LANDSCHAFT***           <208>
3010 POKE 53280,2:POKE 53281,14                  <041>
3020 PRINT"{6DOWN,BROWN,RVSON}£¥"                <073>
3030 PRINT"{RVSON,SPACE,BLACK}¥{BROWN,SPACE}¥"
                         <179>
3040 PRINT"{BLACK,RVSON,4SPACE,BROWN}¥"          <222>
3050 PRINT"{BLACK,RVSON,5SPACE,BROWN,2SPACE}¥"
                         <232>
3060 PRINT"{BLACK,RVSON,2SPACE,BROWN,2SPACE,
   BLACK,2SPACE}¥{BROWN,SPACE}¥"                 <246>
3070 PRINT"{RVSON,SPACE,BLACK,4SPACE,BROWN,
   4SPACE}¥"                                     <252>
3080 PRINT"{RVSON,2SPACE,BLACK,6SPACE,BROWN,
   4SPACE}¥";                                    <066>
3090 PRINT"{RVOFF,BLUE}TTTTTTTTTTTTTTTTTTTTTTTT
   TTT";                                         <024>
3100 PRINT"{BLUE}TT TTTTTT TT";                  <189>
3110 PRINT"{GREY 2,RVSON}£{12SPACE}¥{RVOFF}";
                         <014>
3120 PRINT"{BLUE,2SPACE}TTTTTTT TTT"             <150>
3130 PRINT"{2SPACE}TTTT TTT{SPACE,GREY 2,RVSON}
   £{14SPACE}¥{RVOFF,BLUE,3SPACE}TTT TTTT"       <240>
3140 PRINT" TTTT{3SPACE}T{2SPACE,GREY 3}
   OPOPOPOPOPOPOPOP{BLUE}TT{3SPACE}TTTTT"        <131>
3150 PRINT"TTTTTTT{3SPACE}TTT TTTTT TTTTTTTT
   {3SPACE}TTTT{2SPACE}TTT";                     <140>
3160 PRINT:PRINT"{3SPACE}TTTTTTTT{5SPACE}TTTT
   {3SPACE}TTTTTTTTT{2SPACE}TTTTT"               <255>
3170 PRINT:PRINT"TTTTTT {3SPACE}TTTTTTTTT{6SPACE}
   TTTTTTTTTTTTTT"                               <239>
3200 RETURN                                      <026>
4000 DATA 0,62,0     :REM SONNE                  <158>
4001 DATA 0,255,192 :REM (READ Q8)               <178>
4002 DATA 3,255,224                              <243>
4003 DATA 7,255,240                              <246>
4004 DATA 15,255,248                             <046>
4005 DATA 31,255,252                             <040>
4006 DATA 31,255,252                             <041>
4007 DATA 63,255,252                             <047>
4008 DATA 63,255,254                             <050>
4009 DATA 63,255,254                             <051>
4010 DATA 63,255,254                             <052>
4011 DATA 63,255,254                             <053>
4012 DATA 63,255,254                             <054>
4013 DATA 31,255,252                             <048>
4014 DATA 31,255,252                             <049>
4015 DATA 31,255,252                             <050>
4016 DATA 15,255,248                             <058>
4017 DATA 7,255,240                              <004>
4018 DATA 3,255,224                              <003>
4019 DATA 1,255,192                              <006>
4020 DATA 0,62,0                                 <102>
4500 DATA 0,0,0 :REM HELIKOPTER                  <208>
4501 DATA 0,0,0 :REM   (READ Q)                  <152>
4502 DATA 21,85,85                               <191>
4503 DATA 0,4,0                                  <023>
4504 DATA 0,12,0                                 <071>
4505 DATA 64,63,0                                <136>
4506 DATA 64,255,0                               <188>
4507 DATA 255,250,0                              <234>
4508 DATA 255,242,128                            <087>
4509 DATA 64,242,192                             <039>
4510 DATA 64,240,192                             <038>
4511 DATA 0,63,192                               <192>
4512 DATA 0,63,0                                 <085>
4513 DATA 0,12,0                                 <080>
4514 DATA 0,4,16                                 <089>
4515 DATA 1,85,64                                <151>
4516 DATA 0,,,,,,,,,,,,,,,                       <208>
4600 DATA 0,0,0                                  <116>
4601 DATA 0,0,0   :REM (READ Q6)                 <050>
4602 DATA 0,0,0                                  <118>
4603 DATA 0,0,0                                  <119>
4604 DATA 2,0,128                                <229>
4605 DATA 0,68,0                                 <183>
4606 DATA ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
   ,,,,,,,                                       <034>
4700 DATA 0,0,0                                  <217>
4701 DATA 0,0,0 :REM (READ Q7)                   <152>
4702 DATA 0,0,0                                  <219>
4703 DATA 0,0,0                                  <220>
4704 DATA 0,130,0                                <065>
4705 DATA 0,0,0                                  <222>
4706 DATA 1,17,0                                 <024>
4707 DATA ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
   ,,,                                           <004>
4810 DATA 12,2,0,4,0,0,192:REM MOTOR             <182>
4820 DATA 16,2,0,6,0,0,64 :REM (READ X)          <200>
4830 DATA 0,5,14,31                              <037>
5000 DATA 0,0,0 :REM MANN NACH RECHTS            <220>
5001 DATA 0,0,0 :REM     (HAND OBEN)             <096>
5002 DATA 0,0,0 :REM     (READ Q1)               <192>
5003 DATA 0,0,0                                  <009>
5004 DATA 0,12,0                                 <061>
5005 DATA 0,14,0                                 <064>
5006 DATA 0,8,0                                  <020>
5007 DATA 0,31,192                               <173>
5008 DATA 0,40,0                                 <066>
5009 DATA 0,72,0                                 <072>
5010 DATA 0,136,0                                <122>
5011 DATA 0,8,0                                  <025>
5012 DATA 0,24,0                                 <072>
5013 DATA 0,36,0                                 <076>
5014 DATA 0,66,128                               <187>
5015 DATA 0,33,0                                 <075>
5016 DATA ,,,,,,,,,,,,,,,                        <150>
5500 DATA 0,0,0 :REM MANN NACH RECHTS            <210>
5501 DATA 0,0,0 :REM     (HAND UNTEN)            <188>
5502 DATA 0,0,0 :REM     (READ Q2)               <183>
5503 DATA 0,0,0                                  <255>
5504 DATA 0,12,0                                 <051>
5505 DATA 0,14,0                                 <054>
5506 DATA 0,8,0                                  <010>
5507 DATA 0,28,0                                 <061>
5508 DATA 0,42,0                                 <058>
5509 DATA 0,25,0                                 <060>
5510 DATA 0,24,128                               <167>
5511 DATA 0,8,0                                  <015>
5512 DATA 0,8,0                                  <016>
5513 DATA 0,8,0                                  <017>
5514 DATA 0,8,0                                  <018>
5515 DATA 0,12,0                                 <062>
5516 DATA 0,0,0                                  <012>
5517 DATA 0,0,0                                  <013>
5518 DATA 0,0,0                                  <014>
5519 DATA 0,0,0                                  <015>
5520 DATA 0,0,0                                  <016>
6000 DATA 0,0,0 :REM MANN STEHT                  <109>
6001 DATA 0,0,0 :REM (READ Q2)                   <172>
6002 DATA 0,0,0                                  <244>
6003 DATA 0,0,0                                  <245>
6004 DATA 0,12,0                                 <041>
6005 DATA 0,14,0                                 <044>
6006 DATA 0,8,0                                  <000>
6007 DATA 0,24,0                                 <047>
6008 DATA 0,44,0                                 <050>
6009 DATA 0,78,0                                 <058>
6010 DATA 0,134,0                                <100>
6011 DATA 0,7,128                                <111>
6012 DATA 0,4,0                                  <002>
6013 DATA 0,8,0                                  <007>
6014 DATA 0,8,0                                  <008>
6015 DATA 0,12,0                                 <052>
6016 DATA ,,,,,,,,,,,,,,,                        <130>
7000 DATA 146,0,0                                <073>
7001 DATA 0,32,0   :REM (READ Q3)                <206>
7002 DATA 0,4,0                                  <228>
7003 DATA 0,1,0                                  <226>
7004 DATA 0,0,0                                  <226>
7005 DATA 0,0,64                                 <029>
7006 DATA 0,0,0                                  <228>
7007 DATA 0,0,0                                  <229>
7008 DATA 0,0,16                                 <029>
7009 DATA 0,0,0                                  <231>
```



Die Diskette für eine Ausgabe kostet **29,90*** Mark. Sie werden bei einigen Disketten bestimmte Programme vermissen. Deren Autoren konnten sich nicht entschließen, ihr Programm im Rahmen des Leserservice für eine Verbreitung auf Datenträger freizugeben. Bei den Ausgaben 5 und 6 können noch Kassetten (VC ...) bestellt werden. Zu den Programmen sind immer die Seitenzahlen angegeben, unter der Sie die Beschreibungen in der entsprechenden Ausgabe finden können. Der Diskette liegen also keinerlei Informationen bei. Lesen Sie daher aufmerksam die Anleitung (ob SYS-Befehle nötig sind, in welcher Reihenfolge geladen werden muß, eventuelle Sprach- oder Speichererweiterungen und ähnliches mehr) in dem jeweiligen Artikel nach. Aus Aktualitätsgründen wird jeweils die abgedruckte Version angeboten. **Eventuelle systematische Fehler, die sich noch im Programm befinden können, müssen von Ihnen selbst, nach Studium des Druckfehlerteufelchens, korrigiert werden.**

Ausgabe 2/85

Bestell-Nr. L 6 8502A — DM 29,90*

Commodore 64

Checksummer 64
MSE
SMON (Teil 4)
Grab des Pharao (LdM)
Basic-Lader
MAC
RAM-Floppy
Scrolling
Notlandung
Super-Memory

VC 20

Checksummer VC 20
Familienplanung (AdM)
Super 8-Steuerung
Q+Bert (3K)

Ausgabe 1/85

Bestell-Nr. L 6 8501A — DM 29,90*

Commodore 64	
Checksummer 64	S.72
Handballtrainer (AdM)	S.53
SMON (Teil 3)	S.69
Hi-Eddi (LdM)	S.57
Hypra-Load mal vier	S.82
Tips und Tricks	S.87
Provic 64	S.76
Eingabe (UPB)	S.156
VC 20	
Checksummer VC 20	S.72

Ausgabe 12/84

Bestell-Nr. CB 022 — DM 29,90*

Commodore 64	
Synthesizer (AdM)	S.51
SMON (2. Teil)	S.60
3D-Vier gewinnt	S.96
Trace	S.76
Stringy	S.88
Lader	S.92
Auto	S.84
Listschutz	S.85
Simons Axo (SB)	S.64
Kreuzworträtsel	S.150
VC 20	
Mathematikal Basic (8K>) (LdM)	S.55
Fast Tape	S.80

Ausgabe 11/84

Bestell-Nr. CB 020 — DM 29,90*

Commodore 64	
Turtle Grafik (LdM)	S.48
Schachmeister (AdM)	S.50
SMON (1. Teil)	S.59
Floppykurs	S.117
FPLOT-Befehlserweiterung	S.73
Get Koala pic	S.66
Interrupttechnik	S.84
Exsort (UPB)	S.154
Einzeiler	S.158
Simons Basic Befehlserweiterung (SB)	S.90
VC 20	
Pseudosprites (8K)	S.76
Laterna Magica (8K)	S.68
Betriebssystem-Erweiterung (24K>)	S.88
Supergrafik (GV)	S.71
VC 20-Kurs (GV>)	S.126

Ausgabe 10/84

Bestell-Nr. CB 019 — DM 29,90*

Commodore 64	
Finanzmathematik (AdM)	S.68
Hypra-Load (LdM)	S.67
Hardcopy Compact 2	S.86
Hardcopy MPS 801	S.82
Hardcopy VC 1526 neu	S.83
Hardcopy Gemini-10X	S.85
Hardcopy FX-80	S.88
Hardcopy VC 1520 farbig	S.84
Apocalypse now	S.106
Supercopy	S.102
Disk-Dump	S.95
Diskettenorganisation	S.97
User-Port-Tastatur	S.92
Maske-(UPB)	S.172
VC 20	
Epedemic	S.112
Video-Vorspann	S.81

Ausgabe 9/84

Bestell-Nr. CB 014 — DM 29,90*

Commodore 64	
Indexsequentielle Adreßdatei	S.54
Spring Vogel (LdM)	S.68
Orgel/Synthesizer (AdM)	S.70
Sprite Aid +	S.89
Screen Change	S.94
List-Stop	S.97
Renew, Datawandler	S.102
Synthetische suchen	S.104
Geregelter Zahlungsverkehr	S.164
VC 20	
Schiebung (GV>)	S.77
Deuzei (8K>)	S.79
Hardcopy 1520 (GV>)	S.87
RS232-Interface (GV>)	S.100
Datawandler (GV>)	S.102

Ausgabe 8/84

Bestell-Nr. CB 013 — DM 29,90*

Commodore 64	
Castle of Doom	S.66
Pac-Boy	S.89
Kopplung	S.73
User-Port-Display	S.97
RS232-Test	S.77
View BAM	S.99
Görlitz Hardcopy	S.83
Milchvieh	S.156
VC 20	
Kudiplo (3K)	S.86
Print at Restore n (GV)	S.101

Ausgabe 7/84

Bestell-Nr. CB 017 — DM 29,90*

Commodore 64	
Terminalprogramm	S.24
Softwarekatalog	S.72
Russvok (SB)	S.76
Crown No. 1	S.80
Space Invaders	S.81
1520 Hardcopy	S.108
Centronics Interface	S.110
Kurvendiskussion	S.116
Copy Rel. Files	S.132
Autostart	S.138
Strubs (OP u. QP)	S.154
VC 20	
Rätsel	S.122

Ausgabe 6/84

Bestell-Nr. CB 018 — DM 29,90*

Commodore 64	
Lehrerkalender	S.64
Morsetrainer	S.72
Supervoc	S.69
Grafische Darstellung (SB)	S.82
Hot Wheels	S.92
VC 20	
Bestell-Nr. VC 008	DM 29,90*
Movemaster (8K)	S.78
Ghost Manor (GV)	S.104
Logic Disass. (3K>)	S.108
Underground (LdM 16K)	S.120

Ausgabe 5/84

Bestell-Nr. CB 016 — DM 29,90*

Commodore 64	
Adreß-&Telefonregister	S.64
Fahrsimulator	S.82
Schatzsucher (LdM)	S.90
VC 20	
Bestell-Nr. VC 007	DM 29,90*
Relative Datei (8K)	S.69
Schmatzer (GV)	S.76
3D-Grafik (8K)	S.78

* Alle Preise inklusive Mehrwertsteuer. Der Versand erfolgt mit offener Rechnung zuzüglich Porto und Verpackung.

Fehlende Hefte erhalten Sie bei: Markt & Technik Vertrieb 64'er Hans-Pinsel-Str. 2, 8013 Haar

Bedeutung der Abkürzungen

*LdM = Listing des Monats
*AdM = Anwendung des Monats
*SB = Simons Basic
*GV = Grundversion
*GV> = alle Speicherversionen können verwendet werden (einschließlich GV)
*3K = 3-KByte-Speichererweiterung wird benötigt
*8K> = Speichererweiterung größer als 8 KByte wird benötigt
*UPB = Unterprogrammbibliothek

Bestellungen richten Sie bitte an: M&T Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar bei München

```
7010 DATA 0,0,0                                   <232>
7011 DATA 0,0,4                                   <237>
7012 DATA 0,0,0                                   <234>
7013 DATA 0,0,0                                   <235>
7014 DATA 0,0,2                                   <238>
7015 DATA ,,,,,,,,,,,,,,,,,                       <241>
7500 DATA 0,0,0 :REM MANN NACH LINKS              <098>
7501 DATA 0,0,0 :REM    (HAND UNTEN)              <148>
7502 DATA 0,0,0 :REM    (READ Q4)                 <145>
7503 DATA 0,0,0                                   <215>
7504 DATA 0,24,0                                  <014>
7505 DATA 0,56,0                                  <020>
7506 DATA 0,8,0                                   <226>
7507 DATA 0,28,0                                  <021>
7508 DATA 0,26,0                                  <020>
7509 DATA 0,25,0                                  <020>
7510 DATA 0,24,128                                <127>
7511 DATA 0,24,0                                  <021>
7512 DATA 0,8,0                                   <232>
7513 DATA 0,8,0                                   <233>
7514 DATA 0,8,0                                   <234>
7515 DATA 0,24,0                                  <025>
7516 DATA,,,,,,,,,,,,,,,                          <100>
8000 DATA 0,0,0 :REM MANN NACH LINKS              <088>
8001 DATA 0,0,0 :REM    (HAND OBEN)               <036>
8002 DATA 0,0,0 :REM    (READ Q5)                 <136>
8003 DATA 0,0,0                                   <205>
8004 DATA 0,24,0                                  <004>
8005 DATA 0,56,0                                  <010>
8006 DATA 0,8,0                                   <216>
8007 DATA 1,255,192                               <170>
8008 DATA 0,8,0                                   <218>
8009 DATA 0,8,0                                   <219>
8010 DATA 0,8,0                                   <220>
8011 DATA 0,8,0                                   <221>
8012 DATA 0,12,0                                  <009>
8013 DATA 0,18,0                                  <016>
8014 DATA 0,161,0                                 <064>
8015 DATA 0,66,0                                  <021>
8016 DATA,,,,,,,,,,,,,,,                          <090>
8090 REM *** END 1.TEIL ***                       <168>
8100 DATA,,,,,,,,,,,,,,,,,,,,,,,,,                <190>
8101 DATA,42,                                     <005>
8102 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,       <200>
8200 DATA,,,,,,,,,,,,,,,,,,,,,,,,,                <127>
8201 DATA,42,,,42,                                <052>
8202 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,       <013>
8300 DATA,,,,,,,,,,,,,,,,,,,,,,,,,                <227>
8301 DATA,221,128,,221,128                        <044>
8302 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,       <113>
8499 REM FLIEGERLIED ******************           <243>
8500 DATA 13,10,250,9,196,500                     <126>
8510 DATA 14,162,100,13,10,100                    <163>
8520 DATA 12,78,100,13,10,250                     <135>
8530 DATA 9,196,500                               <188>
8540 DATA 14,162,100,13,10,100                    <193>
8550 DATA 12,78,100,13,10,250                     <165>
8560 DATA 14,162,100,16,109,100                   <017>
8570 DATA 17,103,100,16,109,250                   <031>
8580 DATA 14,162,125,19,137,750                   <059>
8590 DATA 21,237,250,21,237,125                   <059>
8600 DATA 17,103,250,14,162,500                   <062>
8610 DATA 19,137,250,19,137,125                   <091>
8620 DATA 16,109,250,13,10,500                    <030>
8630 DATA 13,10,125,14,162,125                    <041>
8640 DATA 16,109,250,14,162,125                   <110>
8650 DATA 13,10,125,14,162,250                    <060>
8660 DATA 13,10,125,12,78,125                     <027>
8670 DATA 13,10,750                               <059>
8990 REM *** END 2.TEIL ***                       <049>
9000 DATA,,,,,,,,,,,,,,,,,,,,,,,,                 <194>
9001 DATA 1,213,128,1,157,64,1,213,128            <043>
9002 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,          <204>
10000 DATA,,,,,,,,,,,,,,,,,,,,,                   <042>
10001 DATA 7,162,224,4,50,144,7,42,144,4,38,144,
   7,162,224                                      <147>
10002 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,             <052>
10100 DATA,,,,,,,,,,,,,,,,,,                      <010>
10101 DATA 31,65,120,16,97,68,16,81,68,30,73,68,
   16,69,68,16,67,68,31,65,120                    <151>
10102 DATA,,,,,,,,,,,,,,,,,,,,,,,,                <020>
10200 DATA,,,,,,,,,,,,,,                          <234>
10201 DATA 126,65,62,64,65,33,64,97,33,64,81,33,
   124,73,33                                      <114>
10202 DATA 64,69,33,64,67,33,64,65,33,126,65,62
                   <139>
10203 DATA,,,,,,,,,,,,,,,,,,,,                    <245>
```

»Notlandung«, das lustige Programm (Schluß)

Programmierwettbewerb: Terminalprogramm

Um den Akustikkoppler oder das Modem voll auszunutzen, braucht man ein gutes »Werkzeug«. Nicht alle der meist teuren Terminalprogramme sind ihr Geld wert. Wir setzen 1000 Mark für das beste Terminalprogramm aus.

Was ist ein »gutes« Terminalprogramm, beziehungsweise, was zeichnet dieses Programm gegenüber anderen aus?

Zuerst muß es einwandfrei funktionieren. Es soll für die Kommunikation von Computern über die Telefonleitung sorgen. Darüber hinaus muß das Programm aber komfortabel und benutzerfreundlich sein. In diesem Punkt sind Ihren Ideen keine Grenzen gesetzt. Hierzu einige Anregungen:

Die Übertragungsparameter sollten frei einzustellen sein. Ideal wäre es, wenn ein Prüfprogramm mit eingearbeitet ist, das automatisch die Parametereinstellung des anderen Computers überprüft und sich selbst einstellt.

Parallel zum Empfang sollten die Texte gespeichert werden können (die »Time out«-Regelung bei manchen Mailboxen ist zu beachten!). Diese Texte müssen später auf dem Bildschirm und dem Drucker abrufbar sein. Eigene Nachrichten sollte man vorab schreiben und später als Gesamtheit senden können.

Ein wichtiger Punkt, den man nicht vergessen darf, ist die Übertragung von Programmen.

Gängige Befehle sollten auf den Funktionstasten liegen.

Schicken Sie Ihre Lösung unter dem Stichwort »Programmierwettbewerb: Terminalprogramm« an folgende Adresse:
Markt & Technik Verlag AG
Redaktion 64'er,
Hans-Pinsel-Str. 2,
8013 Haar bei München.
Einsendeschluß: 28. 2. 1985

Fortsetzung von Seite 41

selbst übersetzen. Denn schließlich läuft der Compiler, da in Basic geschrieben, auf jedem Basic-Computer. Als Ergebnis erhält man den Compiler selbst, aber in Maschinensprache. Nun kann der ursprüngliche Compiler erweitert und verbessert werden. Die Menge der Basic-Befehle, die übersetzt werden können, kann ausgeweitet werden. Mit dem frisch übersetzten »alten« Compiler übersetzt man den erweiterten Compiler. Nun hat man schon einen Compiler zur Verfügung, der einiges mehr kann als die erste Version. Aber auch diesen Compiler kann man verbessern, erweitern und schließlich wieder übersetzen und so fort.

So kann man aus einem primitiven, sogenannten Tiny-Basic-Compiler einen sehr komplexen Basic-Compiler »züchten«. Der große Vorteil, einen Compiler in seiner eigenen Quellsprache zu schreiben, liegt also darin, daß der Compiler selbst als erstes Programm von jeder Erweiterung des Sprachumfanges profitiert. Dieser Vorgang, der auf den ersten Blick ein wenig an Münchhausen erinnert, der sich am eigenen Schopfe aus dem Sumpf zog, wird als »Bootstrapping« bezeichnet.

Wie bei allen rekursiven Verfahren bleibt noch die Frage nach dem Anfangsschritt offen. Denn man braucht mindestens einen ganz einfachen und primitiven ersten Compiler für das Bootstrapping.

Nun, im Falle eines Basic-Compilers ist die Lösung einfach. Da der Compiler selbst in Basic geschrieben ist, kann er auch vom Basic-Interpreter ausgeführt werden. Bei anderen Sprachen muß man in den sauren Apfel beißen und die erste Version »von Hand« in Basic übersetzen, damit das Ding überhaupt erst mal läuft. (ev)

Inserentenverzeichnis

Der Schweizer Ausgabe liegen Prospekte der Firma Techn. Lehrinstitut Onken bei.

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael M. Pauly (py)
Stellv. Chefredakteur: Michael Scharfenberger (sc)
Redakteure: aa = Albert Absmeier, leitender Redakteur, ev = Volker Everts, gk = Georg Klinge, hm = Harald Meyer, rg = Christian Rogge
Redaktionsassistenz: Gerda Siegl (202)
Fotografie: Janos Feitser, Titelfoto: Alex Kempkens

Layout: Leo Eder (Ltg.), Dagmar Berninger, Willi Gründl

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Alpenstrasse 14, CH-6300 Zug, Tel. 042-223155/56, Telex: 862329 mut ch
USA: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. (415) 424-0600; Telex 752351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Herstellung: Klaus Buck (180)

Anzeigenverkauf: Brigitta Fiebig (211)

Anzeigenverwaltung und Disposition: Michaela Hörl (171)

Anzeigenformate: 1/1-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297 x 210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.

Anzeigenpreise: Es gilt die Anzeigenpreisliste Nr. 2 vom 1. Januar 1985.
Anzeigengrundpreise: 1/1 Seite sw: DM 8500,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1400,-. Vierfarbzuschlag DM 3800,-. Plazierung innerhalb der redaktionellen Beiträge: Mindestgröße 1/2-Seite
Anzeigen im Computer-Markt: Die ermäßigten Preise im Computer-Markt gelten nur innerhalb des geschlossenen Anzeigenteils, der ohne redaktionelle Beiträge ist. 1/1-Seite sw: DM 6400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. **Anzeigen in der Fundgrube:** **Private Kleinanzeigen** mit maximal 5 Zeilen Text DM 5,- je Anzeige.
Gewerbliche Kleinanzeigen: DM 11,- je Zeile Text.
Auf alle Anzeigenpreise wird die gesetzliche MwSt. jeweils zugerechnet.

Vertriebsleitung, Werbung: Hans Hörl (114)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebsgesellschaft mbH, Hauptstätterstraße 96, 7000 Stuttgart 1, Telefon (07 11) 6483-0

Erscheinungsweise: 64'er, Magazin für Computerfans, erscheint monatlich, Mitte des Vormonats.

Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-119. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement verlängert sich zu den dann jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Ablauf schriftlich gekündigt wird.

Bezugspreise: Das Einzelheft kostet DM 6,50. Der Abonnementspreis beträgt im Inland DM 78,- pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,- für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z.B. USA) um DM 38,-, in Ländergruppe 2 (z.B. Hongkong) um DM 58,-, in Ländergruppe 3 (z.B. Australien) um DM 68,-.

Druck: E. Schwend GmbH, Schmollerstr. 31, 7170 Schwäbisch Hall

Verantwortlich: Für redaktionellen Teil: Michael M. Pauly.
Für Anzeigen: Hannelore Schmidt.

Redaktions-Direktor: Michael Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052

Mitglied der Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e.V. (IVW), Bad Godesberg.
ISSN 0344-8843



Telefon-Durchwahl im Verlag:

***Wählen Sie direkt:** Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.*